Introduction to FHE and the TFHE Scheme

Ilaria Chillotti
KU Leuven & Zama

Lattices: From Theory to Practice
Simons Institute
30 April 2020

Table of Contents

Table of Contents

Homomorphic Encryption

Allows to perform computations on encrypted messages, without decrypting.

$$\begin{array}{ccc} \boxed{m_1} & & \boxed{m_1} +_{hom} \boxed{m_2} = \boxed{m_1 + m_2} \\ & \longrightarrow & \\ \boxed{m_2} & & \boxed{m_1} \times_{hom} \boxed{m_2} = \boxed{m_1 \times m_2} \end{array}$$

- Possibly any function
- Different message spaces
- Secret and public key solutions

Many applications

- Computations over sensitive data (medical, biological, financial, etc.)
- Outsourced computations
- Electronic voting
- Multiparty Computations
- And more...

Once upon a time...

- **1978** - Rivest, Adleman, Dertouzos: *privacy homomorphisms*
- . . .
- **2009** - Gentry: first **fully** homomorphic encryption construction

What happened in the meantime?

Many schemes are homomorphic...

- RSA
- ElGamal
- ...
- Paillier
- Goldwasser-Micali

...but only **partially**.

Some schemes can support both addition and multiplication, but "with limits":

- **somewhat**: example the scheme by Boneh, Goh and Nissim 2005
- **leveled**.........

A world full of noise...

Example: [DGHV10]

Scheme based on the Approximate GCD problem [HG01], proposed by Van Dijk, Gentry, Halevi, Vaikuntanathan in 2010.

$$\boxed{c = m + 2r + pq}$$

- $m \in \{0, 1\}$ message
- $p \in \mathbb{Z}$ secret key
- $q \in \mathbb{Z}$ large ($p \ll q$)
- $r \in \mathbb{Z}$ small noise ($r \ll p$)

To decrypt: ciphertext modulo p and then modulo 2.

A world full of noise...

$$c_1 = m_1 + 2r_1 + pq_1 \qquad c_2 = m_2 + 2r_2 + pq_2$$

Addition (XOR):

$$c_1 + c_2 = (m_1 + m_2) + 2(r_1 + r_2) + p(q_1 + q_2)$$

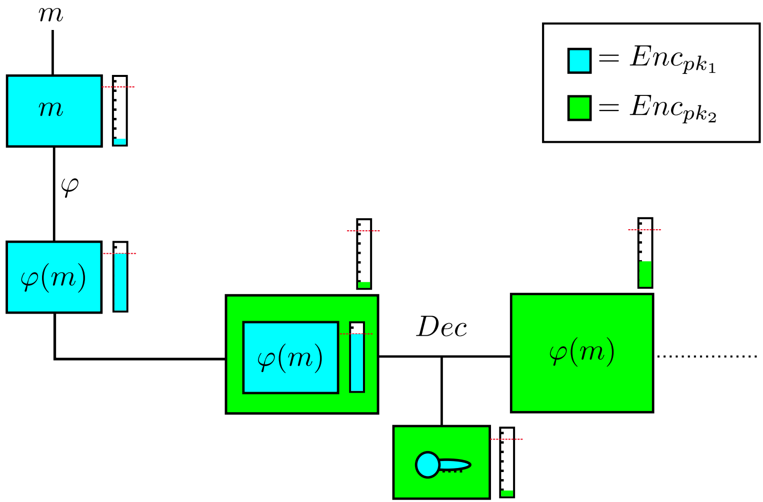
Noise amount : double...

Multiplication (AND):

$$c_1 \cdot c_2 = (m_1 \cdot m_2) + 2(2r_1 \cdot r_2 + \ldots) + p(q_1 \cdot q_2 + \ldots)$$

Noise amount : square...

If noise grows too much, a correct decryption cannot be guaranteed!


m
m
φ
$\varphi(m)$
$\varphi(m)$
Dec
$\varphi(m)$
$= Enc_{pk_1}$
$= Enc_{pk_2}$

Bootstrapping

Bootstrapping is very costly!

"To bootstrap, or not to bootstrap, that is the question" (semi cit.)

Leveled homomorphic

Set the function, there exist parameters to homomorphically evaluate it.

- ✔ Fast evaluations
- ✘ The depth has to be known in advance

Fully homomorphic

Set the parameters, it is possible to homomorphically evaluate any function.

- ✘ Slow evaluations (**Bootstrapping**)
- ✔ No depth limitations

Existing schemes

Lattice problems

Approximate-GCD [HG01], **NTRU** [HPS98], **(Ring-)LWE** [Reg05],[SSTX09],[LPR10]

- In this workshop we will mainly concentrate on (Ring-)LWE-based solutions

Some (Ring-)LWE-based schemes

"BGV-like"

- B(G)V: [BV11], [BGV12]
- B/FV: [Bra12], [FV12]
- HEAAN: [CKKS17]

"GSW-like"

- GSW: [GSW13]
- FHEW: [DM15]
- TFHE: [CGGI16-17]

In practice, they are less different than expected: Chimera [BGGJ19]

Some implementations

- cuFHE
- FHEW
- HEAAN
- HElib
- Lattigo
- Microsoft SEAL
- NFLlib
- nuFHE
- Palisade
- TFHE
- ...

Table of Contents

FHEW

[DM15]

- GSW-based construction
- **They build a FHE brick: a bootstrapped NAND gate**
- Slow (but significantly improved): $\sim$ 0.69 seconds per bootstrapped NAND gate
- Large bootstrapping keys: $\sim$ 1 GByte

[DM15]: L. Ducas, D. Micciancio, *FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second*, EUROCRYPT 2015

TFHE

Bootstrapped versions [CGGI16]

- Slow (but significantly improved):
  ~~~ 0.69~~ ~ 0.05 seconds per bootstrapped NAND gate
- Slow (but significantly improved) [CGGI17]:
  ~~~ 0.69~~ ~~~ 0.05~~ ~ 0.013 seconds per bootstrapped NAND gate
- ~~Large~~ bootstrapping keys: ~~~ 1 GByte~~ ~ 23.4 MBytes

Leveled versions [CGGI17]

- Fast(er) for small depth circuits
- New techniques to improve leveled evaluations
- New Bootstrapping for larger circuits

[CGGI16]: I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, *Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds*, ASIACRYPT 2016
[CGGI17]: I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, *Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE*, ASIACRYPT 2017

Torus

$(\mathbb{T}, +, \cdot)$ is a $\mathbb{Z}$-module
(the external product $\cdot : \mathbb{Z} \times \mathbb{T} \to \mathbb{T}$ is well defined)

- ✔ It is an abelian group: $x + y \mod 1$, $-x \mod 1$, ...
- ✔ It is a $\mathbb{Z}$-module: $0 \cdot \frac{1}{2} = 0$ is defined!
- ✘ It is not a Ring: $0 \times \frac{1}{2}$ is **not** defined!

Torus polynomials

$(\mathbb{T}_N[X], +, \cdot)$ is a $\mathfrak{R}$-module

- Here, $\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$
- And $\mathbb{T}_N[X] = \mathbb{T}[X] \mod (X^N + 1)$

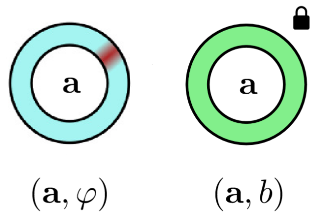
TFHE ciphertexts

LWE

Message $\mu \in \mathbb{T}$, secret key $\mathbf{s} \in \mathbb{B}^n$

$$\boxed{\mathbf{c} = (\mathbf{a}, b) \in \mathbb{T}^{n+1}}$$

- $\mathbf{a}$ random mask, $b = \mathbf{s} \cdot \mathbf{a} + \varphi$
- $\varphi = e + \mu$, $e \in \mathbb{T}$ Gaussian



$\mathbb{T} = \mathbb{R} \mod 1$, $\mathbb{B} = \{0, 1\}$

RLWE

Message $\mu \in \mathbb{T}_N[X]$, secret key $s \in \mathbb{B}_N[X]$

$$\boxed{\mathbf{c} = (a, b) \in \mathbb{T}_N[X]^2}$$

- a random mask, $b = s \cdot a + e + \mu$, $e \in \mathbb{T}_N[X]$ Gaussian

$\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1) \mod 1$, $\mathbb{B}_N[X] = \mathbb{Z}[X]/(X^N + 1)$ with binary coefs

TFHE operations

RGSW

Message $m \in \mathbb{Z}_N[X]$, secret key $\mathbf{s} \in \mathbb{B}_N[X]$ as in RLWE

$$\boxed{C = Z + m \cdot G_2 \in \mathbb{T}_N[X]^{2\ell \times 2}}$$

- with Z is a list of 2ℓ RLWE encryptions of 0
- with G_2 the **gadget** matrix

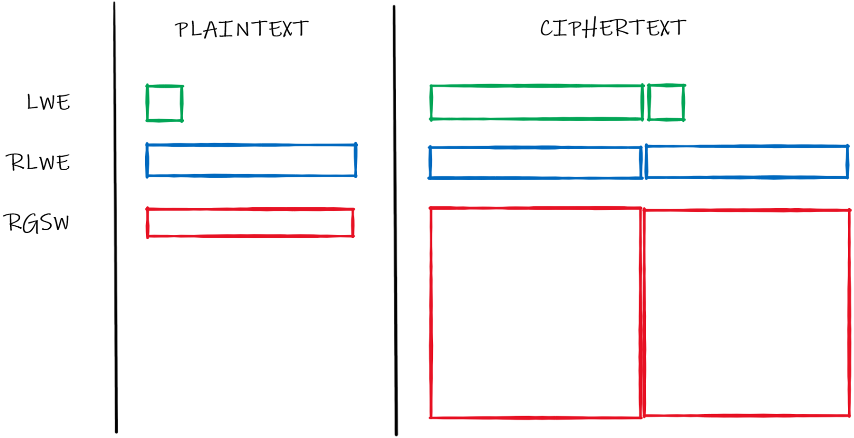
$$G_2 = \left(\begin{array}{c|c} \mathbf{g} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{g} \end{array}\right), \text{ with } \mathbf{g}^T = (2^{-1}, ..., 2^{-\ell})$$

G_2^{-1}: easy to decompose $\mathbb{T}_N[X]$ elements w.r.t. G_2

$\mathbb{Z}_N[X] = \mathbb{Z}[X]/(X^N + 1)$

TFHE ciphertexts

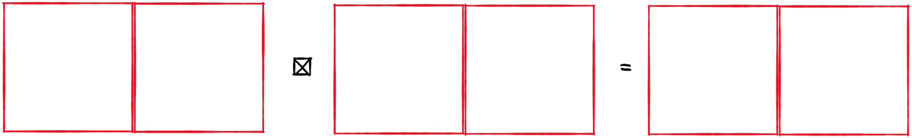
	plaintext	ciphertext	linear combinations	product
LWE	$\mathbb{T}$	$\mathbb{T}^{n+1}$	✔	✘
RLWE	$\mathbb{T}_N[X]$	$\mathbb{T}_N[X]^2$	✔	✘
RGSW	$\mathbb{Z}_N[X]$	$\mathbb{T}_N[X]^{2\ell\times 2}$	✔	✔



TFHE products

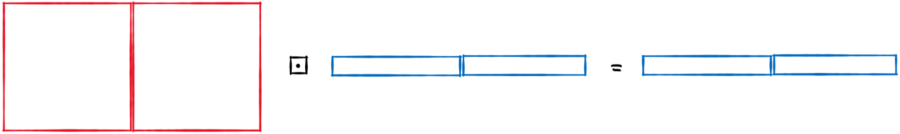
Internal RGSW product

$$C \boxtimes D = G_2^{-1}(D) \cdot C = \begin{bmatrix} G_2^{-1}(\mathbf{d_1}) \cdot C \\ \vdots \\ G_2^{-1}(\mathbf{d_{2\ell}}) \cdot C \end{bmatrix} = \begin{bmatrix} C \boxdot \mathbf{d_1} \\ \vdots \\ C \boxdot \mathbf{d_{2\ell}} \end{bmatrix}$$



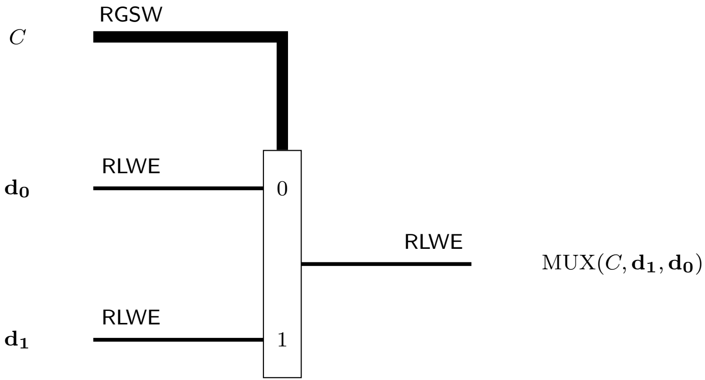
External RGSW – RLWE product [CGGI16],[BP16]

$$C \boxdot \mathbf{d} = G_2^{-1}(\mathbf{d}) \cdot C$$



TFHE MUX

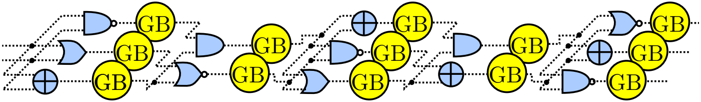
$\mathrm{MUX}(C, \mathbf{d_1}, \mathbf{d_0}) = C \boxdot (\mathbf{d_1} - \mathbf{d_0}) + \mathbf{d_0}$



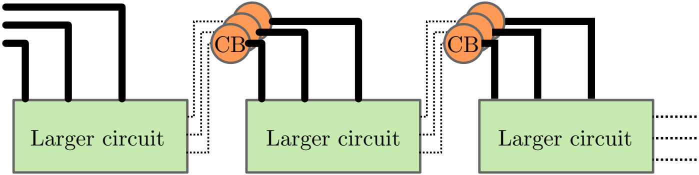
Largely used in TFHE leveled and bootstrapped constructions.

How often shall we bootstrap?

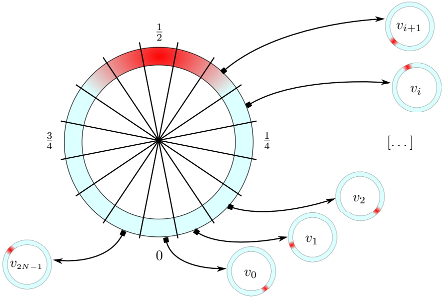
Gate bootstrapping: bootstrap after every gate (like [DM15])



Circuit bootstrapping: bootstrap after a larger circuit



Gate bootstrapping



- Input LWE ciphertext

$$\mathbf{c} = (\mathbf{a}, b)$$

- Depending on

$$\varphi = b - \mathbf{a} \cdot \mathbf{s}$$

we compute an output LWE ciphertext encrypting $v_\varphi \in \mathbb{T}$

Gate bootstrapping

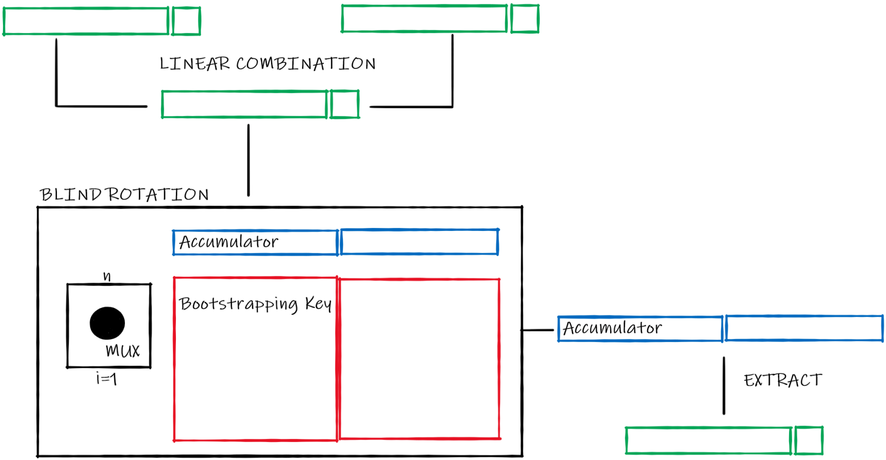
1. Start from (a trivial) RLWE ciphertext of message[a]

$$ACC = v_0 + v_1 X + \cdots + v_{N-1} X^{N-1}$$

2. Do a **blind rotation** of ACC by $-\varphi$ positions (i.e. $ACC \cdot X^{-\varphi}$)
3. **Extract** the constant term of ACC (which encrypts v_φ)

[a] N coefficients modulo $X^N + 1$ can be viewed as $2N$ coefficients modulo $X^{2N} - 1$ s.t. $v_{N+i} = -v_i$

Gate Bootstrapping



Look-Up Table evaluation

The RLWE slots can be used in an optimal way

- LWE: messages $m \in \mathbb{T}$
- RLWE: messages $\mathbf{m} \in \mathbb{T}_N[X]$

$$\mathbf{m} = \sum_{i=0}^{N-1} m_i \cdot X^i \qquad \sim \qquad \mathbf{m} = (m_0, m_1, \ldots, m_{N-1})$$

m_0	m_1	m_2	$\ldots$	m_{N-2}	m_{N-1}

Generally
$N = 2^{10}$

LUT evaluation: Batching and Vertical Packing

LookUp Tables (LUT)

$$f : \mathbb{B}^d \longrightarrow \mathbb{T}^s$$

$$x = (x_0, \ldots, x_{d-1}) \longmapsto f(x) = (f_0(x), \ldots, f_{s-1}(x))$$

Example with $d = 3$ and $s = 2$

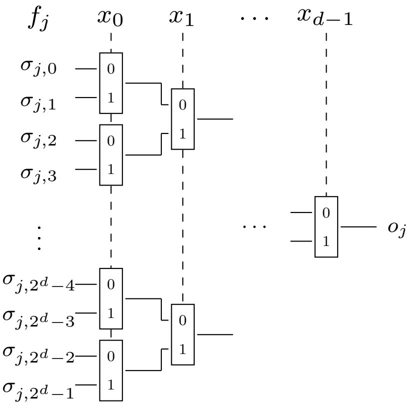
x_0	x_1	x_2	f_0	f_1
0	0	0	0.5	0.3
1	0	0	0.25	0.7
0	1	0	0.1	0.61
1	1	0	0.83	0.9
0	0	1	0.23	0.47
1	0	1	0.67	0.42
0	1	1	0.78	0.12
1	1	1	0.35	0.95

LUT largely used in cryptology (ex. evaluation of arbitrary functions, SBoxes, ...)

LUT evaluation: Batching and Vertical Packing

How to evaluate it?

x_0	$\cdots$	x_{d-1}	f_0	$\cdots$	f_{s-1}
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{s-1,0}$
1	$\cdots$	0	$\sigma_{0,1}$	$\cdots$	$\sigma_{s-1,1}$
0	$\cdots$	0	$\sigma_{0,2}$	$\cdots$	$\sigma_{s-1,2}$
1	$\cdots$	0	$\sigma_{0,3}$	$\cdots$	$\sigma_{s-1,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{0,2^d-4}$	$\cdots$	$\sigma_{s-1,2^d-4}$
1	$\cdots$	1	$\sigma_{0,2^d-3}$	$\cdots$	$\sigma_{s-1,2^d-3}$
0	$\cdots$	1	$\sigma_{0,2^d-2}$	$\cdots$	$\sigma_{s-1,2^d-2}$
1	$\cdots$	1	$\sigma_{0,2^d-1}$	$\cdots$	$\sigma_{s-1,2^d-1}$

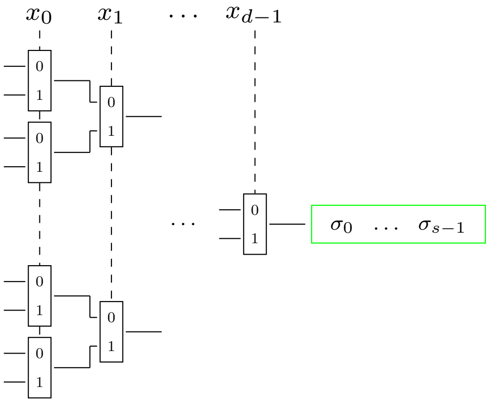


LUT evaluation: Batching and Vertical Packing

Batching (Horizontal Packing)

- Pack the outputs in a RLWE ciphertext (green box)

x_0	$\cdots$	x_{d-1}	f_0	$\cdots$	f_{s-1}
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{s-1,0}$
1	$\cdots$	0	$\sigma_{0,1}$	$\cdots$	$\sigma_{s-1,1}$
0	$\cdots$	0	$\sigma_{0,2}$	$\cdots$	$\sigma_{s-1,2}$
1	$\cdots$	0	$\sigma_{0,3}$	$\cdots$	$\sigma_{s-1,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{0,2^d-4}$	$\cdots$	$\sigma_{s-1,2^d-4}$
1	$\cdots$	1	$\sigma_{0,2^d-3}$	$\cdots$	$\sigma_{s-1,2^d-3}$
0	$\cdots$	1	$\sigma_{0,2^d-2}$	$\cdots$	$\sigma_{s-1,2^d-2}$
1	$\cdots$	1	$\sigma_{0,2^d-1}$	$\cdots$	$\sigma_{s-1,2^d-1}$



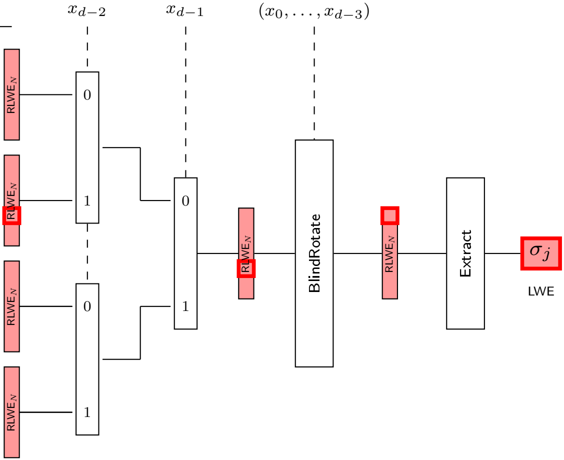
LUT evaluation: Batching and Vertical Packing

Vertical packing

x_0	$\cdots$	x_{d-1}	f_0	$\cdots$	f_{s-1}
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{s-1,0}$
1	$\cdots$	0	$\sigma_{0,1}$	$\cdots$	$\sigma_{s-1,1}$
0	$\cdots$	0	$\sigma_{0,2}$	$\cdots$	$\sigma_{s-1,2}$
1	$\cdots$	0	$\sigma_{0,3}$	$\cdots$	$\sigma_{s-1,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{0,2^d-4}$	$\cdots$	$\sigma_{s-1,2^d-4}$
1	$\cdots$	1	$\sigma_{0,2^d-3}$	$\cdots$	$\sigma_{s-1,2^d-3}$
0	$\cdots$	1	$\sigma_{0,2^d-2}$	$\cdots$	$\sigma_{s-1,2^d-2}$
1	$\cdots$	1	$\sigma_{0,2^d-1}$	$\cdots$	$\sigma_{s-1,2^d-1}$

Vertical Packing

x_0	$\cdots$	x_{d-1}	f_j
0	$\cdots$	0	$\sigma_{j,0}$
1	$\cdots$	0	$\sigma_{j,1}$
0	$\cdots$	0	$\sigma_{j,2}$
1	$\cdots$	0	$\sigma_{j,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{j,2^d-4}$
1	$\cdots$	1	$\sigma_{j,2^d-3}$
0	$\cdots$	1	$\sigma_{j,2^d-2}$
1	$\cdots$	1	$\sigma_{j,2^d-1}$



Mix them all...

- Depending on the use case, choose which type of packing is the best
- You can mix them: they are compatible

x_0	$\cdots$	x_{d-1}	f_0	$\cdots$	f_{s-1}
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{s-1,0}$
1	$\cdots$	0	$\sigma_{0,1}$	$\cdots$	$\sigma_{s-1,1}$
0	$\cdots$	0	$\sigma_{0,2}$	$\cdots$	$\sigma_{s-1,2}$
1	$\cdots$	0	$\sigma_{0,3}$	$\cdots$	$\sigma_{s-1,3}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{0,2^d-4}$	$\cdots$	$\sigma_{s-1,2^d-4}$
1	$\cdots$	1	$\sigma_{0,2^d-3}$	$\cdots$	$\sigma_{s-1,2^d-3}$
0	$\cdots$	1	$\sigma_{0,2^d-2}$	$\cdots$	$\sigma_{s-1,2^d-2}$
1	$\cdots$	1	$\sigma_{0,2^d-1}$	$\cdots$	$\sigma_{s-1,2^d-1}$

More TFHE

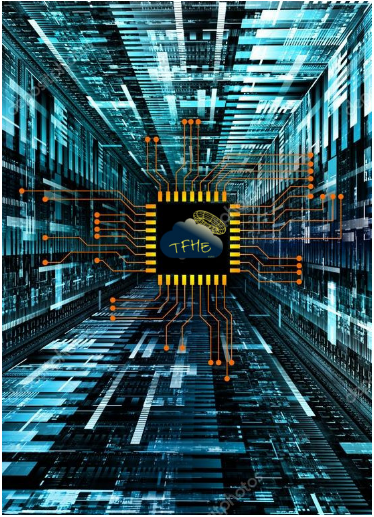
Seen in this presentation

- Basic construction
- Gate bootstrapping
- Evaluation of LUT (leveled)

More...

- Evaluate deterministic (weighted) finite automata
- The homomorphic counter TBSR
- Circuit bootstrapping
- ...

TFHE implementation



TFHE: Fast Fully Homomorphic Encryption over the Torus

- Open source C/C++ library
  `https://tfhe.github.io/tfhe/`
- Distributed under Apache 2.0 license

Gate bootstrapping

- All gates implemented in the official release

Circuit bootstrapping and leveled operations

- Implemented in the experimental repository
  `https://github.com/tfhe/experimental-tfhe`

TFHE in Gate Bootstrap mode versus Circuit Bootstrap mode

TFHE Gate Bootstrapping

- Input/Output: LWE $\rightarrow$ LWE
- Gate bootstrapping in 10-20 ms
- All binary gates have the same cost

TFHE Circuit Bootstrapping

- Input/Output: LWE $\rightarrow$ RGSW
- Circuit bootstrapping in 137 ms
- After many transitions 34 μs

Evaluate about 70 bootstrapped binary gates per second.

Evaluate a LUT from 16-bit input to 8-bit output in 1 second.

Bit Overhead

- LWE: 2.46 KB (encrypts 1 message)
- RLWE: 8 KB (encrypts up to 1024 messages)
- RGSW: 48 KB (encrypts up to 1024 messages)

- Implementation tested on (single core) Intel i7 and Intel i9 processor laptops
- Parameters have 128-bits of security according to the LWE estimator

Table of Contents

Conclusion

Some TFHE related works

- GPU implementations: cuFHE, nuFHE
- Neural network applications: [BMMP18], TFHE-Chimera solution at iDASH 2019
- Multi-key: MK-TFHE [CCS19]
- Use in MPC: Onion Ring ORAM [CCR19]

Thank you!

Questions?

Bibliography

[RAD78] R. L. Rivest, L. Adleman, M. L. Dertouzos. On data banks and privacy homomorphisms. Foundations of secure computation 1978.
[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. STOC 2009.
[RSA78] R. L. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 1978.
[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory 1985.
[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. EUROCRYPT 1999.
[GM82] S. Goldwasser, S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. STOC 1982.
[BGN05] D. Boneh, E-J. Goh, K. Nissim. Evaluating 2-dnf formulas on ciphertexts. TCC 2005.
[DGHV10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan. Fully homomorphic encryption over the integers. EUROCRYPT 2010.
[HG01] N. Howgrave-Graham. Approximate integer common divisors. CaLC 2001.
[HPS98] J. Hoffstein, J. Pipher, J. H. Silverman. NTRU: A ringbased public key cryptosystem. ANTS-III 1998.
[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC 2005.
[SSTX09] D. Stehlé, R. Steinfeld, K. Tanaka, K. Xagawa. Efficient public key encryption based on ideal lattices. ASIACRYPT 2009.
[LPR10] V. Lyubashevsky, C. Peikert, O. Regev. On ideal lattices and learning with errors over rings. EUROCRYPT 2010.
[BV11] Z. Brakerski, V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. CRYPTO 2011.
[BGV12] Z. Brakerski, C. Gentry, V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. ITCS 2012.

Bibliography

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. CRYPTO 2012.
[FV12] J. Fan, F. Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012.
[CKKS17] J. H. Cheon, A. Kim, M. Kim, Y. Song. Homomorphic encryption for arithmetic of approximate numbers. ASIACRYPT 2017.
[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. CRYPTO 2013.
[DM15] L. Ducas, D. Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. EUROCRYPT 2015.
[CGGI16] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. ASIACRYPT 2016.
[CGGI17] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. ASIACRYPT 2017.
[BGGJ19] C. Boura, N. Gama, M. Georgieva, D. Jetchev. CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes. NutMic 2019.
[BP16] Z. Brakerski, R. Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. CRYPTO 2016.
[BMMP18] F. Bourse, M. Minelli, M. Minihold, P. Paillier. Fast Homomorphic Evaluation of Deep Discretized Neural Networks. CRYPTO 2018.
[CCS19] H. Chen, I. Chillotti, Y. Song. Multi-Key Homomorphic Encryption from TFHE. ASIACRYPT 2019.
[CCR19] H. Chen, I. Chillotti, L. Ren. Onion Ring ORAM: Efficient Constant Bandwidth Oblivious RAM from (Leveled) TFHE. CCS 2019.