Models of Computation for Homomorphic Encryption. TFHE - Chimera

Nicolas Gama and Mariya Georgieva Belorgey

Based on joint work with: C. Boura, I. Chillotti, M. Izabachene, D. Jetchev

Little history of FHE

SANDBOXAQ ⊕inpher

Constant-time Privacy-preserving computations

- FHE programs behave essentially like circuits.
- 2009: Bootstrapping (evaluate one homomorphic NAND)
- The overhead between plaintext circuit and FHE is $O(1)$ in time and memory

42: That's it!! CQFD?

- 2009: First bootstrapping = hours of computations
- 2009-now: Just wait for scientific progress... and brace for impact!!

Little history of FHE

SANDBOXAQ™ ⊕inpher

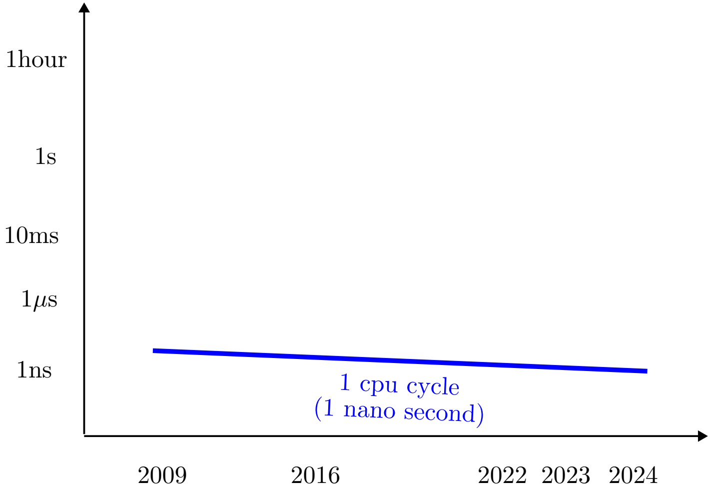
Constant-time Privacy-preserving computations

- FHE programs behave essentially like circuits.
- 2009: Bootstrapping (evaluate one homomorphic NAND)
- The overhead between plaintext circuit and FHE is $O(1)$ in time and memory

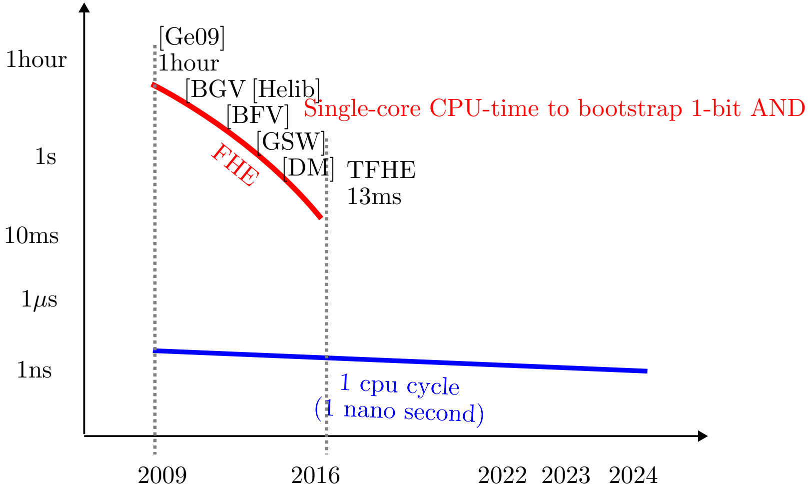
42: That's it!! CQFD?

- 2009: First bootstrapping = hours of computations
- 2009-now: Just wait for scientific progress... and brace for impact!!

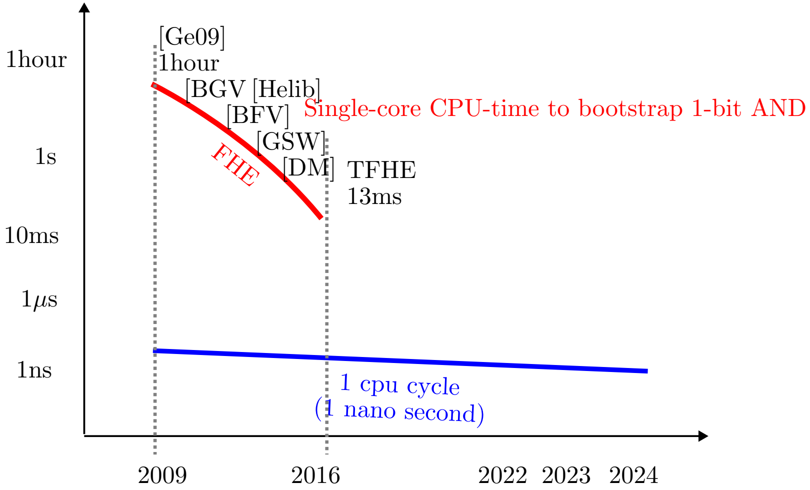
Bootstrapping: the beginnings



Bootstrapping: the beginnings



Bootstrapping: the beginnings



Properties of a gate-bootstrapping ciphertext

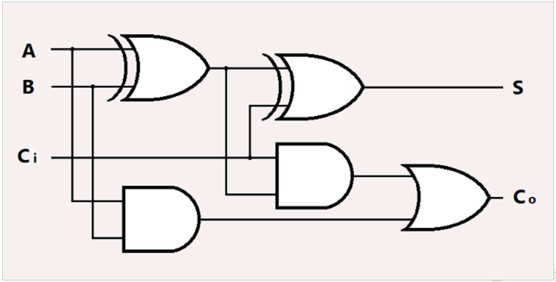
SANDBOXAQ™ inpher

Properties of a gate-bootstrapping ciphertext

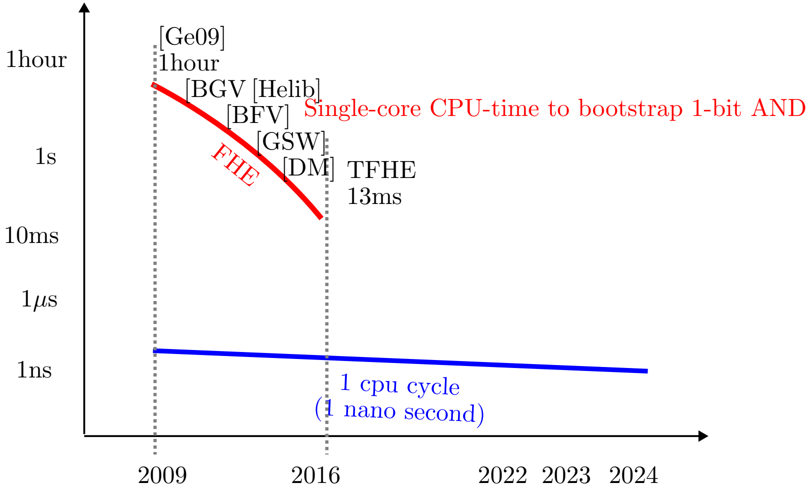
- a message = one bit

Composition rules: gates

- constant gates (0,1)
- unary gates (copy, not)
- binary gates (and, or, nand, nor, ...)
- selector gate (mux)

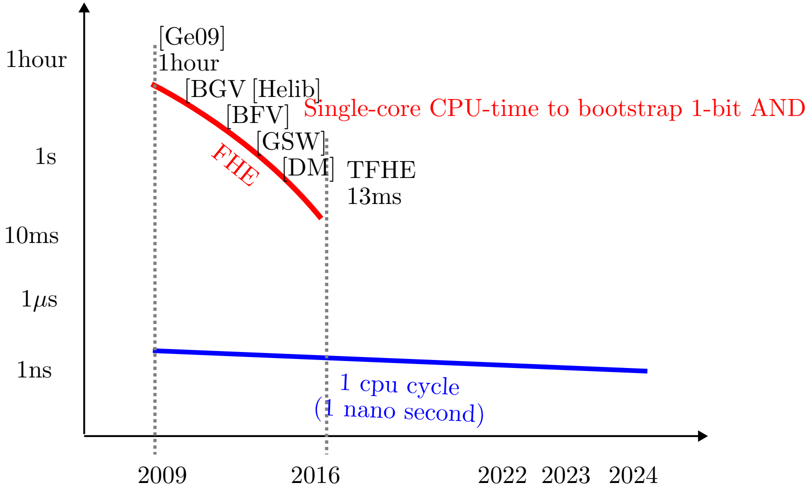


Little history: are we there yet?

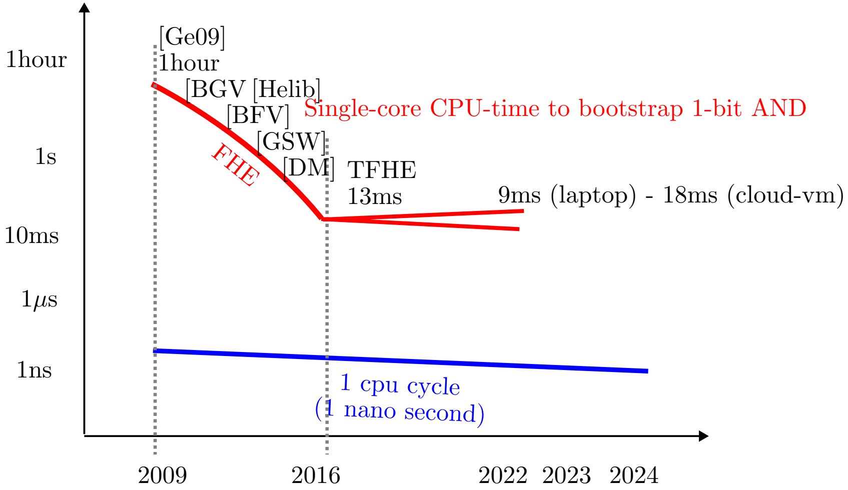


Little history: are we there yet?

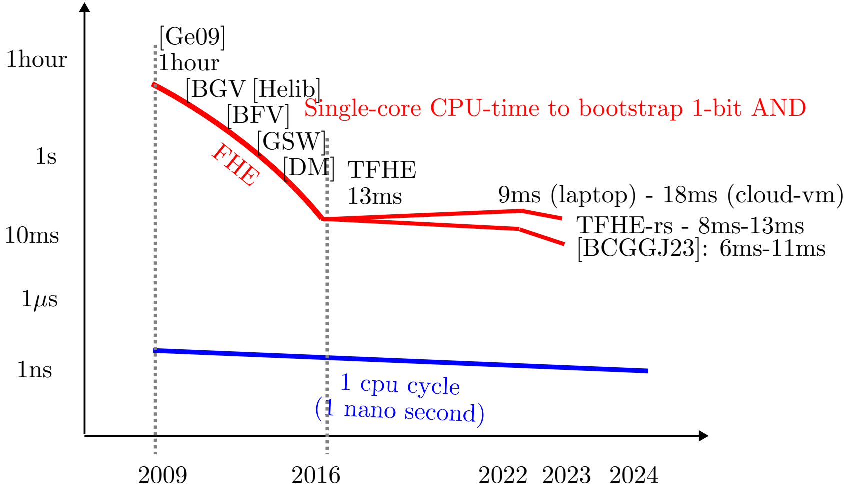
SANDBOXAQ inpher



Little history: are we there yet?



Little history: are we there yet?



Gate bootstrapping: pros/cons

SANDBOXAQ ⊕inpher

Pros of gate bootstrapping

- circuits are standard and easy to generate
- asymptotically $O(1)$ optimal w.r.t. plaintext circuit

Cons of gate bootstrapping

- the $O(1)$ theoretical overhead factor is huge in practice
  - **timing** 10ms vs. 1 nanosec per cycle (vs. 1 picosec physical),
  - **size** 20kB ciphertext per bit.
  - **parallelization** 100 gates in parallel vs. billion gates in parallel.

The reality: 2023

- Only use-cases that take a fraction of second in plaintext are feasible via only Gate Bootstrapping.
- Practical FHE requires a plan B!

Gate bootstrapping: pros/cons

Pros of gate bootstrapping

- circuits are standard and easy to generate
- asymptotically $O(1)$ optimal w.r.t. plaintext circuit

Cons of gate bootstrapping

- the $O(1)$ theoretical overhead factor is huge in practice
  - **timing** 10ms vs. 1 nanosec per cycle (vs. 1 picosec physical),
  - **size** 20kB ciphertext per bit.
  - **parallelization** 100 gates in parallel vs. billion gates in parallel.

The reality: 2023

- Only use-cases that take a fraction of second in plaintext are feasible via only Gate Bootstrapping.
- Practical FHE requires a plan B!

Plan B: LHE to the rescue of FHE

SANDBOXAQ ⊕inpher

FHE Fully Homomorphic Encryption ($O(1)$ from optimal)

$\exists$ crypto params s.t. $\forall$ circuit C, we can evaluate C homomorphically.

LHE Leveled Homomorphic Encryption (not $O(1)$ from optimal)

$\forall$ circuit C, $\exists$ crypto params s.t. we can evaluate C homomorphically.

Chimera: Using FHE to boost LHE

- FHE and LHE are not mutually exclusive, they should be used together!
- Some LHE schemes are much faster for massive low multiplication-depth arithmetic in practice (integer (BFV) or FP (CKKS)).
- Other LHE schemes are quite good for evaluating automata (RGSW, TFHE).
- Bootstrapping is quite good at evaluating LUTs and univariate non-linear functions, like conversions.

Compilation

SANDBOXAQ™ ⊕inpher

Nice to have

Let the user write the desired program as a high-level pseudocode.

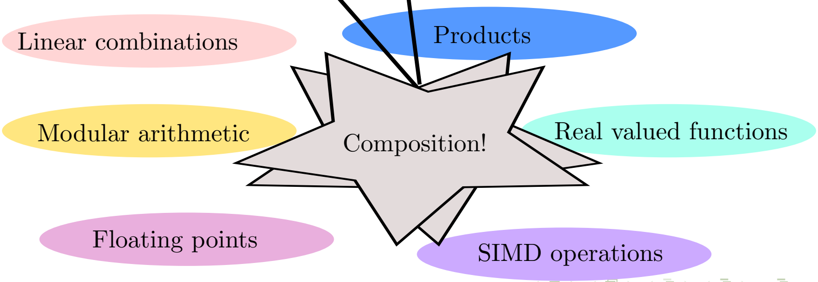
```
def myfunction(x, y, z, bigvector, bigmatrix)
    a := 2 * x + 3 * y * z mod 15
    c := 30 * cos(a / 8.)
    return c * bigmatrix * bigvector
```

Compilation

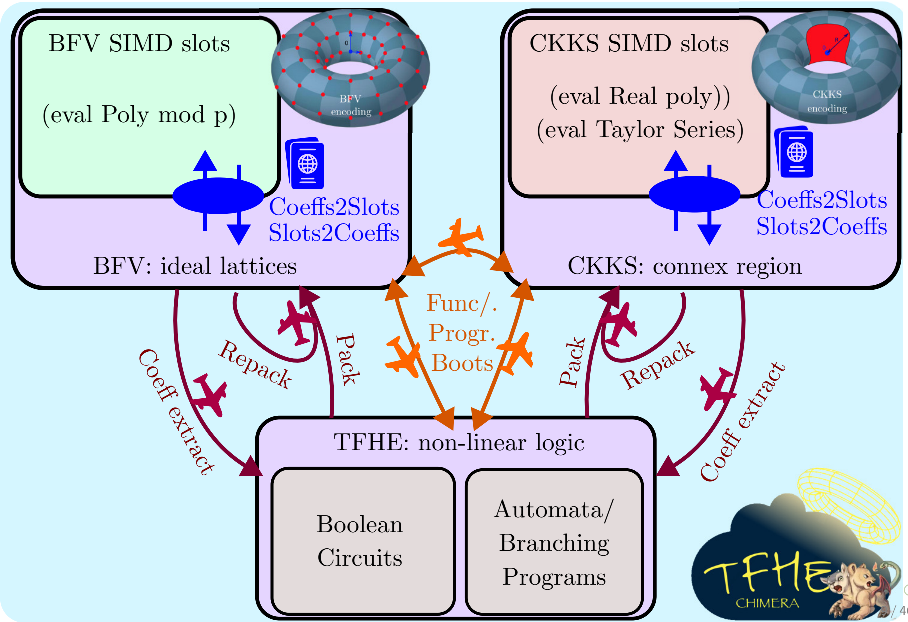
Nice to have

Let the user write the desired program as a high-level pseudocode.

```
def myfunction(x, y, z, bigvector, bigmatrix)
    a := 2 * x + 3 * y * z mod 15
    c := 30 * cos(a / 8.)
    return c * bigmatrix * bigvector
```



Chimera World Map



Properties of a leveled ciphertext

Properties of a leveled HE ciphertext

- a message (encoded in a polynomial)
  - one bit? one integer?
  - a vector of integers mod p?
  - a vector of floats?
- a homomorphic budget: 3 equivalent definitions
  - noise rate: $0 < \alpha < 1$
  - homomorphic budget: $-\log_2(\alpha) \geq 0$ points
  - homomorphic level: 1 level $\approx$ 30 points

Homomorphic budget (a.k.a. level, noise-rate)

Noise rate and homomorphic budget

A noise rate $a < 1$ corresponds to a Homomorphic budget of $-\log_2(\alpha) > 0$ points, and quantifies the number of homomorphic operations that can be carried out on a FHE ciphertext.

	efficiency	HE operations
$-\log_2(\alpha) \approx 0$	small ciphertext, small key, fast operations	exhausted
$-\log_2(\alpha) \approx 30$	native 32-bit arithmetic	1 multiplications
$-\log_2(\alpha) \approx 60$	native 64-bit arithmetic	2 multiplications
$-\log_2(\alpha) \approx 300$	slow 300-bit arithmetic	10 multiplications

Rule of thumb

- more points = more homomorphic power. But bigger ciphertexts, larger keys, larger arithmetic, slower operations.
- decreasing the homomorphic budget points is easy (*"modulus switching/rescaling"*)
- increasing it is much harder (*"bootstrapping"*).

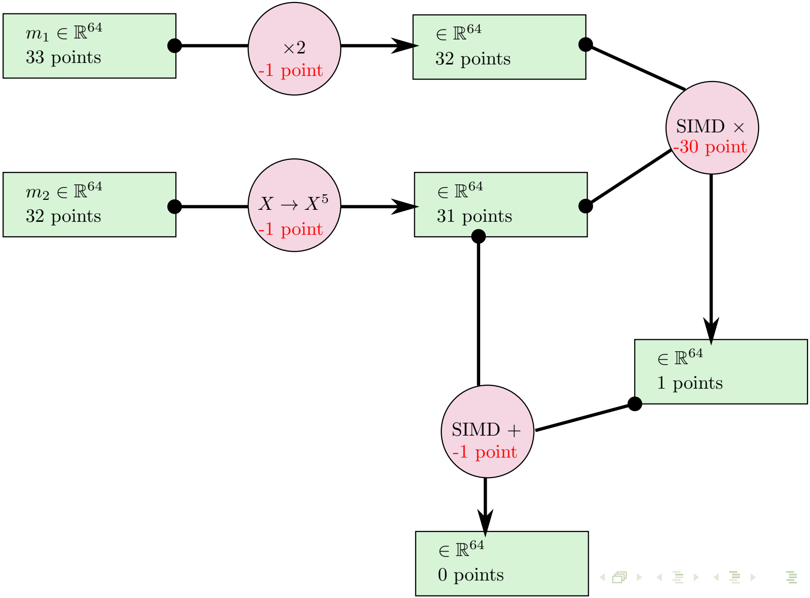
Operations and composition rules

Arithmetic circuit model of computation

A graph of polynomial arithmetic operations. Each operation impact the noise (so the homomorphic budget):

Operation	ciphertext type	homomorphic budget impact
slot-wise product:	RLWE	-30 points
sum	RLWE	-1 point
public rotation:	RLWE	-0 point
linear combinations $\sum e_i \mathbf{c}_i$:	RLWE	$-\log(1 + \|e\|_1)$ points
substitution with X^t:	RLWE	-1 point

Operations and composition rules

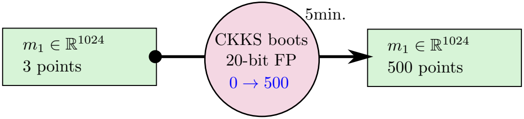


Bootstrapping rules

Arithmetic Circuit: bootstrapping

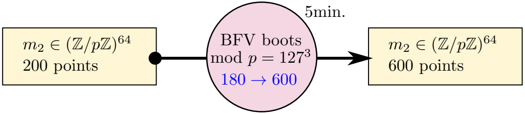
- **Input:** a ciphertext with (nearly) depleted budget
- **Output:** a ciphertext of the same message with much larger budget
- **Restrictions/Rules:**
  - Message space, maximal FP precision, modulus.
  - Minimum input level/points, output level/points
  - Running time

Bootstrapping: example in the litterature



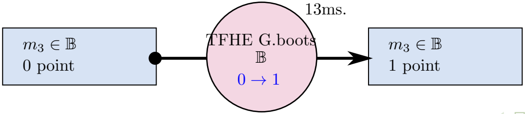
Bootstrapping is per type of message space: real, mod, binary.

1 Bootstrapping proposes a fixed input/output level



Bootstrapping may use all the slots, or just a fewer number.

Bootstrapping may also require a minimum input level > 0



Arithmetic circuits: Pros and Cons

SANDBOXAQ™ ⊕inpher

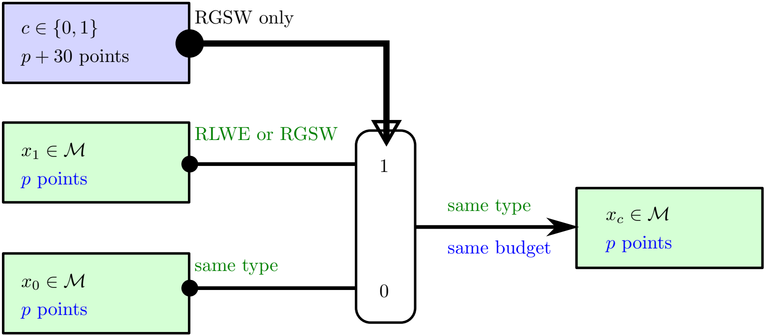
Arithmetic circuits: Pros

- the *"assembly language"* of LWE: no loss, no overhead
- good for executing SIMD arithmetic use-cases
- Newest bootstrapping have usually a fast amortized time per slot.

Arithmetic circuits: Cons

- General use-cases are hard to convert to polynomial arithmetic circuits. (think CPU vs. GPU)
- The rare use-cases that work are already described "in assembly"

RGSW-based private selector circuits



data Input and output have the same homomorphic budget!

LUT evaluation

LookUp Tables (LUT) to evaluate arbitrary functions:

$$f : \mathbb{B}^d \longrightarrow \mathbb{T}^s$$
$$x = (x_0, \ldots, x_{d-1}) \longmapsto f(x) = (f_0(x), \ldots, f_{s-1}(x))$$

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

Evaluation via MUX tree

LUT evaluation

LookUp Tables (LUT) to evaluate arbitrary functions:

$$f : \mathbb{B}^d \longrightarrow \mathbb{T}^s$$
$$x = (x_0, \ldots, x_{d-1}) \longmapsto f(x) = (f_0(x), \ldots, f_{s-1}(x))$$

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

Evaluation via MUX tree

LUT evaluation

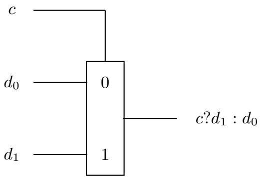
LookUp Tables (LUT) to evaluate arbitrary functions:

$$f : \mathbb{B}^d \longrightarrow \mathbb{T}^s$$
$$x = (x_0, \ldots, x_{d-1}) \longmapsto f(x) = (f_0(x), \ldots, f_{s-1}(x))$$

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

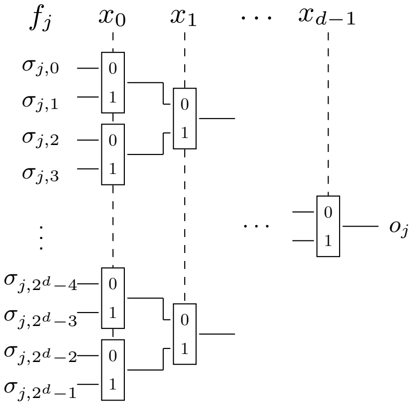
Evaluation via MUX tree



LUT evaluation

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	$\sigma_{0,2^d-1}$	$\cdots$	$\sigma_{s-1,2^d-1}$



DFA versus det-WFA

TFHE 2016: DFA (deterministic finite automata)

- Decisional: returns **accepted** (1) or **rejected** (0)

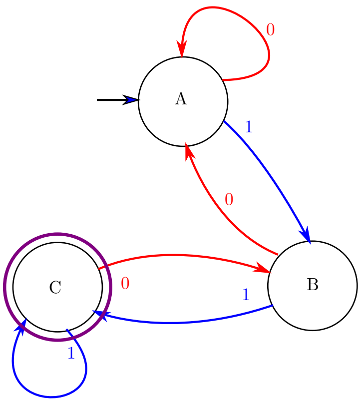
TFHE 2017: det-WFA (deterministic weighted finite automata)

- Computational: returns a **weight** in $\mathbb{T}_N[X]$

Weights act like a "memory" that stores the result all along the evaluation

DFA versus WFA

Deterministic Finite Automata (DFA)

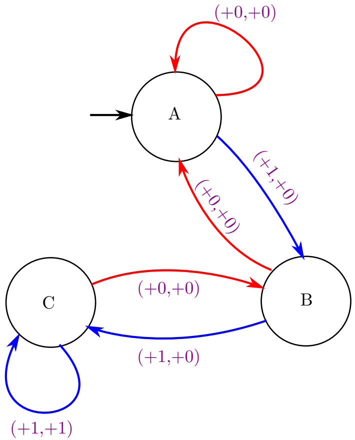


Acceptance

"00101" $\rightarrow$ False

"10111" $\rightarrow$ True

Deterministic Weighted Finite Automata (det-WFA)



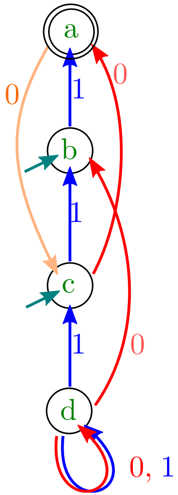
Weight Computation

"00101" $\rightarrow$ (2,0)

"10111" $\rightarrow$ (4,1)

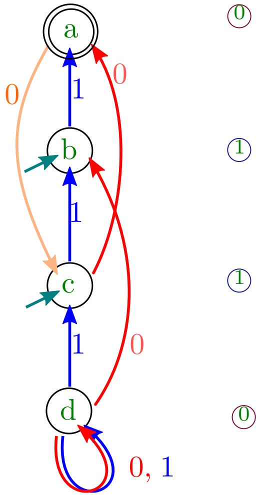
DFA computational models

mirror($\mathcal{L}$)
rev. det. autom.

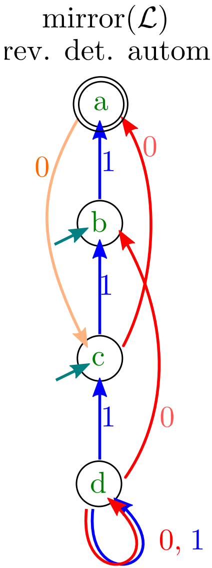


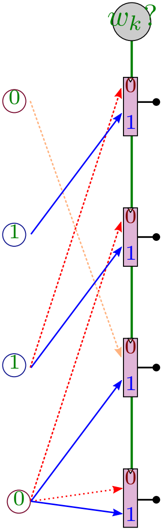
DFA computational models

mirror($\mathcal{L}$)
rev. det. autom.



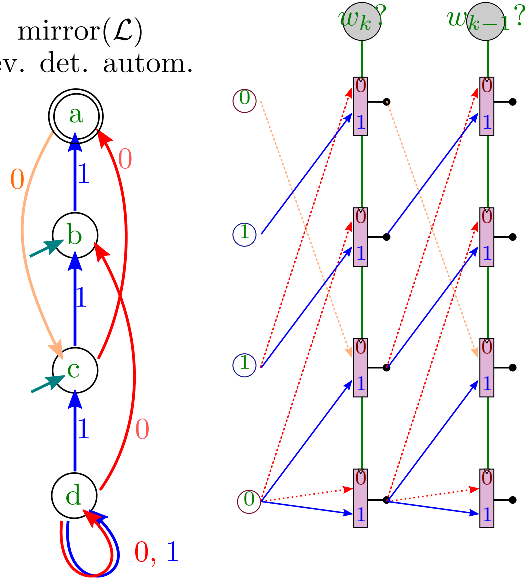
DFA computational models





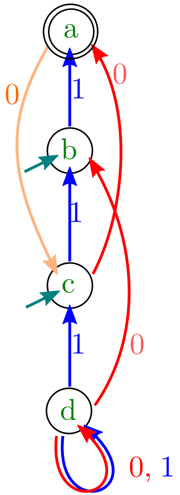
DFA computational models

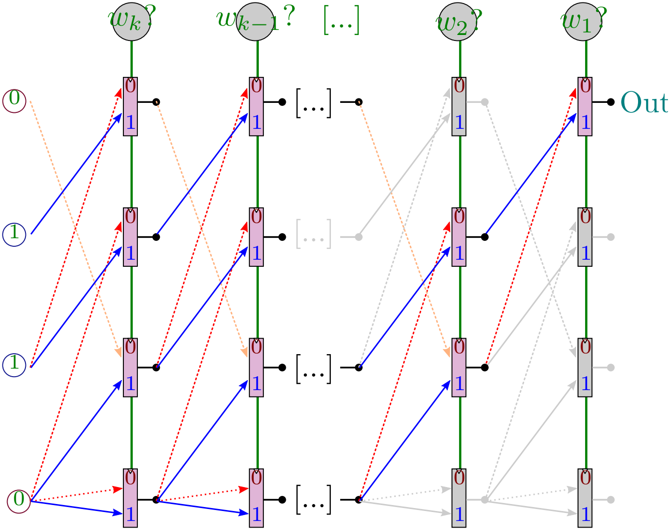
mirror($\mathcal{L}$)
rev. det. autom.



DFA computational models

mirror($\mathcal{L}$)
rev. det. autom.





Computation of the maximum

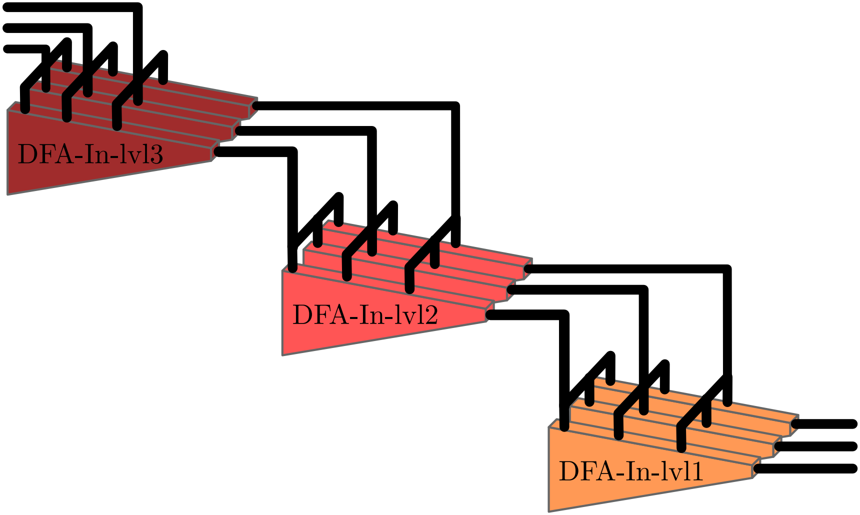
SANDBOXAQ™ ⊕inpher

Example: evaluation of $m = MAX(x, y)$

Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$.
We want to compute $m = (m_1, \ldots, m_n) = MAX(x, y)$.

- **DFA**: evaluate n DFA, one per output bit
- **Det-WFA**: evaluate 1 det-WFA, the result given in a single path

Arbitrary long Composition of automata?

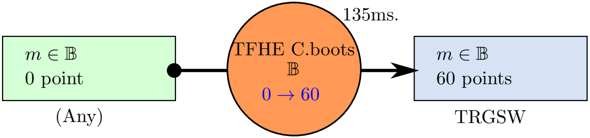


TFHE in Circuit Bootstrap mode

SANDBOXAQ™ inpher

Circuit bootstrapping CGGI2017

- Take advantage that the message space is binary
- And that input/output levels are very low ($0 \to 60$ points)
- Reconstruct a TRGSW encryption directly from its internal structure [CGGI17] rather than as the output of larger homomorphic operations (see [GSW13], [AP14] constructions).



Accepted inputs:

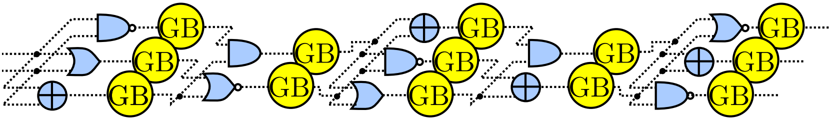
A TLWE ciphertext on binary message space $\{0, \frac{1}{2}\}$

One coefficient of a TRLWE ciphertext over $\{0, \frac{1}{2}\}^N$

One coefficient of a TRGSW ciphertext over $\{0, 1\}^N$

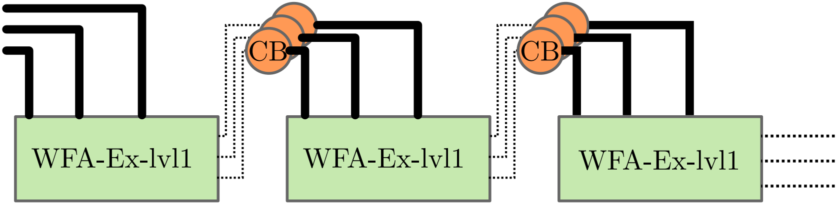
Circuit mode versus Gate Bootstrap mode

Gate bootstrapping mode



versus (or together with?)

Circuit bootstrapping mode



In Summary: The Chimera VM

Logical unit

- Digital circuits
- Lookup Tables
- Deterministic Automata (finite and weighted)

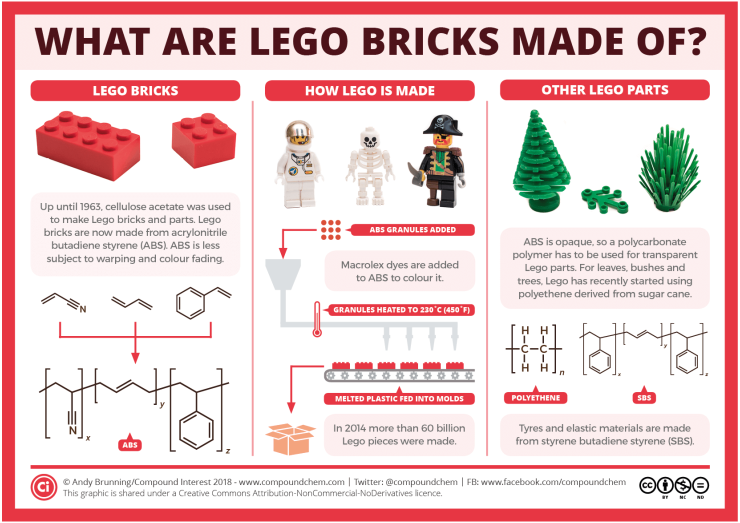
Heavy arithmetic unit

- SIMD fixed-point and modular unit
- Support also convolution, big-integers

Composability, Compilation

- A rich VM capturing all the capabilities of RLWE-based FHE.
- Immediate link to the lattice geometry and its security.
- Is it feasible to compile for this programming model?

Anatomy of a ciphertext



Integer/Real/Complex Polynomials

SANDBOXAQ™ ⊕inpher

Ring of polynomials with coefficients $\in \mathbb{Z}, \mathbb{R}$ or $\mathbb{C}$ mod $X^N + 1$:

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

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

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

Examples (Real): $N = 2$

$$(1.2 + 2.3X) \cdot (3.2 + 4.1X) = 3.84 + 12.28X + 9.43X^2 = 12.28X - 5.59 \mod (X^2 + 1)$$

$(\mathbb{Z}_N[X], +, \times)$, $(\mathbb{R}_N[X], +, \times)$ and $(\mathbb{C}_N[X], +, \times)$ are well defined as rings

- ✔ $(\mathbb{Z}_N[X], +)$, $(\mathbb{R}_N[X], +)$ and $(\mathbb{C}_N[X], +)$ are groups
- ✔ Multiplication $x \times y$ is well-defined!

Torus $\mathbb{T}$ and Torus Polynomials $\mathbb{T}_N[X]$

$\mathbb{T} = \mathbb{R}/\mathbb{Z}$

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

- ✔ It is a group $x + y \mod \mathbb{Z}$, and $-x \mod \mathbb{Z}$
- ✔ It is a $\mathbb{Z}$-module: $3 \cdot 0.6 = 0.8 \mod \mathbb{Z}$ is defined!
- ✘ It is **not** a ring: 0×0.6 is **not** defined!

$\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1) \mod \mathbb{Z}$: polynomials with coeffs $\in \mathbb{R}/\mathbb{Z} \mod X^N + 1$

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

- $(2X + 3) \cdot (0.4X + 0.5) = (0.2X + 0.7) \mod X^2 + 1 \mod \mathbb{Z}$
- external product by integers polynomial

Torus $\mathbb{T}$ and Torus Polynomials $\mathbb{T}_N[X]$

SANDBOXAQ™ inpher

$\mathbb{T} = \mathbb{R}/\mathbb{Z}$

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

- ✔ It is a group $x + y \mod \mathbb{Z}$, and $-x \mod \mathbb{Z}$
- ✔ It is a $\mathbb{Z}$-module: $3 \cdot 0.6 = 0.8 \mod \mathbb{Z}$ is defined!
- ✘ It is **not** a ring: 0×0.6 is **not** defined!

$\mathbb{T}_N[X] = \mathbb{R}[X]/(X^N + 1) \mod \mathbb{Z}$: polynomials with coeffs $\in \mathbb{R}/\mathbb{Z} \mod X^N + 1$

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

- $(2X + 3) \cdot (0.4X + 0.5) = (0.2X + 0.7) \mod X^2 + 1 \mod \mathbb{Z}$
- external product by integers polynomial

TFHE Scheme

Consists of three encryption schemes:

- TLWE ciphertext: $\mu \in \mathbb{T} \mapsto (a, b := \mu + \langle a, s\rangle + e)$, $a \in_R \mathbb{T}^n$, $s \in \{0,1\}^n$
- TRLWE ciphertext: $\mu \in \mathbb{T}_N[X] \mapsto (a, b := \mu + s \cdot a + e)$, $a \in_R \mathbb{T}_N[X]^k$, $s \in \mathbb{B}_N[X]$
- TRGSW ciphertext: encrypts elements of $\mathbb{Z}_N[X]$ with small norm

	message	ciphertext	key	lin. combin.	product
TLWE	$\mathbb{T}$	$\mathbb{T}^{n+1}$	$\mathbb{B}^n$	✔	✘
TRLWE	$\mathbb{T}_N[X]$	$\mathbb{T}_N[X]^{k+1}$	$\mathbb{B}_N[X]^k$	✔	✘
TRGSW	$\mathbb{Z}_N[X]$	vector of TRLWE	$\mathbb{B}_N[X]^k$	✔	✔

1. **Internal TRGSW Product** : $\boxtimes$: TRGSW $\times$ TRGSW $\longrightarrow$ TRGSW
2. **External product** : $\boxdot$: TRGSW $\times$ TRLWE $\longrightarrow$ TRLWE

$$(\mu_A, \mu_{\mathbf{b}}) \longmapsto \mu_A \cdot \mu_{\mathbf{b}}$$
$$(e_A, e_{\mathbf{b}}) \longmapsto \|\mu_A\|_1 \cdot e_{\mathbf{b}} + O(e_A)$$

If $\|\mu_A\|_1 = 1$ the noise propagation is linear!

TFHE Scheme

Consists of three encryption schemes:

- TLWE ciphertext: $\mu \in \mathbb{T} \mapsto (a, b := \mu + \langle a, s \rangle + e)$, $a \in_R \mathbb{T}^n$, $s \in \{0,1\}^n$
- TRLWE ciphertext: $\mu \in \mathbb{T}_N[X] \mapsto (a, b := \mu + s \cdot a + e)$, $a \in_R \mathbb{T}_N[X]^k$, $s \in \mathbb{B}_N[X]$
- TRGSW ciphertext: encrypts elements of $\mathbb{Z}_N[X]$ with small norm

	message	ciphertext	key	lin. combin.	product
TLWE	$\mathbb{T}$	$\mathbb{T}^{n+1}$	$\mathbb{B}^n$	✔	✘
TRLWE	$\mathbb{T}_N[X]$	$\mathbb{T}_N[X]^{k+1}$	$\mathbb{B}_N[X]^k$	✔	✘
TRGSW	$\mathbb{Z}_N[X]$	vector of TRLWE	$\mathbb{B}_N[X]^k$	✔	✔

1. **Internal TRGSW Product** : $\boxtimes$: TRGSW $\times$ TRGSW $\longrightarrow$ TRGSW
2. **External product** : $\boxdot$: TRGSW $\times$ TRLWE $\longrightarrow$ TRLWE

$$(\mu_A, \mu_{\mathbf{b}}) \longmapsto \mu_A \cdot \mu_{\mathbf{b}}$$
$$(e_A, e_{\mathbf{b}}) \longmapsto \|\mu_A\|_1 \cdot e_{\mathbf{b}} + O(e_A)$$

If $\|\mu_A\|_1 = 1$ the noise propagation is linear!

TFHE Scheme

SANDBOXAQ ⊕inpher

Consists of three encryption schemes:

- TLWE ciphertext: $\mu \in \mathbb{T} \mapsto (a, b := \mu + \langle a, s \rangle + e)$, $a \in_R \mathbb{T}^n$, $s \in \{0,1\}^n$
- TRLWE ciphertext: $\mu \in \mathbb{T}_N[X] \mapsto (a, b := \mu + s \cdot a + e)$, $a \in_R \mathbb{T}_N[X]^k$, $s \in \mathbb{B}_N[X]$
- TRGSW ciphertext: encrypts elements of $\mathbb{Z}_N[X]$ with small norm

	message	ciphertext	key	lin. combin.	product
TLWE	$\mathbb{T}$	$\mathbb{T}^{n+1}$	$\mathbb{B}^n$	✔	✘
TRLWE	$\mathbb{T}_N[X]$	$\mathbb{T}_N[X]^{k+1}$	$\mathbb{B}_N[X]^k$	✔	✘
TRGSW	$\mathbb{Z}_N[X]$	vector of TRLWE	$\mathbb{B}_N[X]^k$	✔	✔

1. **Internal TRGSW Product** : $\boxtimes$: TRGSW $\times$ TRGSW $\longrightarrow$ TRGSW
2. **External product** : $\boxdot$: TRGSW $\times$ TRLWE $\longrightarrow$ TRLWE

$$(\mu_A, \mu_{\mathbf{b}}) \longmapsto \mu_A \cdot \mu_{\mathbf{b}}$$
$$(e_A, e_{\mathbf{b}}) \longmapsto \|\mu_A\|_1 \cdot e_{\mathbf{b}} + O(e_A)$$

If $\|\mu_A\|_1 = 1$ the noise propagation is linear!

RLWE internal product ?

Internal product requires to evaluate a polynomail in s:

$$\mu_1 \boxtimes \mu_2 = (b_1 - sa_1)(b_2 - sa_2) = b_1 b_2 - (b_1 a_2 + b_2 a_1)s + a_1 a_2 s^2.$$

The term s^2:

- dedicated relinearization/keyswitch techniques (2011, ...)
- but in fact, TRGSW provides the multiplication by a secret s!

The meaning of $b_1 b_2$, $a_1 a_2$, ...

- lift in $\mathbb{R}_N[X]$
- additional message space restrictions are required to make such product meaningful

Homomorphic operations hierarchy

SANDBOXAQ ⊕inpher

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

Homomorphic operations hierarchy

TRLWE

TRGSW

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

External product
$a.x$ for **secret** a

Homomorphic operations hierarchy

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

TRGSW

External product
$a.x$ for **secret** a

$a \in \{0, 1\}$

cmux (selector)
blindrotate $(\times X^{\text{secret } \nu})$
(automata)

Homomorphic operations hierarchy

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

TRGSW

External product
$a.x$ for **secret** a

$a \in \{0, 1\}$

cmux (selector)
blindrotate $(\times X^{\text{secret } \nu})$
(automata)

TFHE Gates API
individual bits
nand, and,
or, xor, ...
mux

Homomorphic operations hierarchy

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

TRGSW

External product
$a.x$ for **secret** a

$a \in \{0, 1\}$

cmux (selector)
blindrotate $(\times X^{\text{secret } \nu})$
(automata)

$a = s$

polynomials in s
(internal products)

TFHE Gates API
individual bits
nand, and,
or, xor, ...
mux

Homomorphic operations hierarchy

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

TRGSW

External product
$a.x$ for **secret** a

$a \in \{0, 1\}$

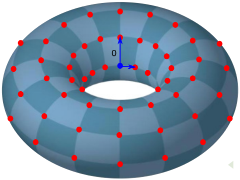
cmux (selector)
blindrotate $(\times X^{\text{secret } \nu})$
(automata)

$a = s$

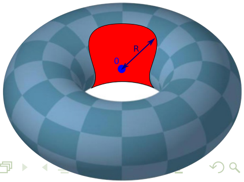
polynomials in s
(internal products)

TFHE Gates API
individual bits
nand, and,
or, xor, ...
mux

Sublattice
(modular ring)



Small Ball
(real ring)



Homomorphic operations hierarchy

SANDBOXAQ ⊕inpher

TRLWE

small integer linear combinations
$x + y$, $x - y$
$a.x$ for public $a \in \mathbb{Z}_N[X]$

TRGSW

External product
$a.x$ for **secret** a

$a \in \{0, 1\}$

cmux (selector)
blindrotate ($\times X^{\text{secret } \nu}$)
(automata)

$a = s$

polynomials in s
(internal products)

Sublattice
(modular ring)

Small Ball
(real ring)

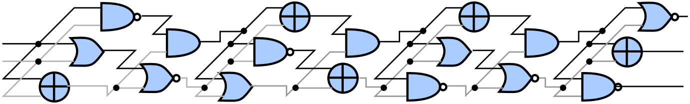
TFHE Gates API
individual bits
nand, and,
or, xor, ...
mux

BFV API
slots mod p
slots add
slots mult
slots rotate

CKKS API
fixed point slots
slots add
slots mult
slots rotate

Different Models of Computations

1 TFHE: Binary circuit evaluation, LUTs, DFAs ...

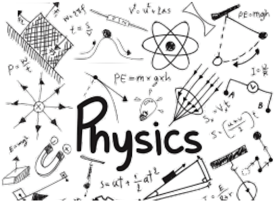


2 B/FV: Integer arithmetic (SIMD)

decimal

$$
\begin{array}{r}
0\ 0\ 1\ 1 \longleftarrow \textit{carries} \\
4\,5\,6\,7 \\
3\,6\,6 \\
\hline
4\,9\,3\,3
\end{array}
$$

3 CKKS: Approximated (fixed-point) computations (SIMD)



Fully Homomorphic Encryption: Chimera [BGGJ20]

Chimera: Combining different FHE schemes: TFHE, B/FV and CKKS

- Unified plaintext space over the Torus
- Switch between ciphertext repsentations (coefficient vs slot packing)

Coefficient and Slot packing

Coefficient packing

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

with $m_i \in \mathbb{C}$ for all $i = 0, 1, \ldots, N-1$

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

Slot packing

$$X^N + 1 = \prod_{i=0}^{N-1} (X - \omega_i) \qquad \sim \qquad \mathbf{m} = (\mathbf{m}(\omega_0), \mathbf{m}(\omega_1), \ldots, \mathbf{m}(\omega_{N-1}))$$

with $\omega_i \in \mathbb{C}$ for all $i = 0, 1, \ldots, N-1$

$\mathbf{m}(\omega_0)$	$\mathbf{m}(\omega_1)$	$\mathbf{m}(\omega_2)$	$\cdots$	$\mathbf{m}(\omega_{N-2})$	$\mathbf{m}(\omega_{N-1})$

Coefficient and Slot packing

Coefficient packing

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

with $m_i \in \mathbb{C}$ for all $i = 0, 1, \ldots, N-1$

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

Slot packing

$$X^N + 1 = \prod_{i=0}^{N-1} (X - \omega_i) \qquad \sim \qquad \mathbf{m} = (\mathbf{m}(\omega_0), \mathbf{m}(\omega_1), \ldots, \mathbf{m}(\omega_{N-1}))$$

with $\omega_i \in \mathbb{C}$ for all $i = 0, 1, \ldots, N-1$

$\boxed{\mathbf{m}(\omega_0)}$ $\boxed{\mathbf{m}(\omega_1)}$ $\boxed{\mathbf{m}(\omega_2)}$ $\cdots$ $\boxed{\mathbf{m}(\omega_{N-2})}$ $\boxed{\mathbf{m}(\omega_{N-1})}$

Morphism between coefficient and slot packing

Morphism

There exists morphism to switch between the coefficient and slot representation! (Vandermonde, DFT,...)

$$VDM = \begin{bmatrix} 1 & \omega_0^1 & \cdots & \omega_0^{N-1} \\ 1 & \omega_1^1 & \cdots & \omega_1^{N-1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \omega_{N-1}^1 & \cdots & \omega_{N-1}^{N-1} \end{bmatrix}.$$

- A complex polynomial $\mod X^N + 1$ carries N complex slots.
- A real polynomial $\mod X^N + 1$ carries $N/2$ complex slots.
- Attention, some additional constraints are needed to define slots for $\mathbb{Z}_N[X]$.

How we can represent all plaintexts over the $\mathbb{T}_N[X]$?

SANDBOXAQ™ ⊕inpher

Ciphertext

(a, b) $\longrightarrow$

$$\mathbb{T}_N[X] + noise?$$

Circuits

$\mathcal{B} = (0, 1)$

Integers

$(\mathbb{Z}/p\mathbb{Z})^n$ $\longrightarrow$

Fixed point

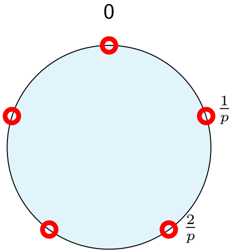
$\mathbb{C}$

BFV scheme (encoding)

- $\mathbb{Z}_N[X] \mod p$: the ring of polynomials with integer $\mod p$ coefficients module $X^N + 1$
- If $X^N + 1$ has N roots mod p, $\mathbb{Z}/p\mathbb{Z}^N$ is isomorphic to $\mathbb{Z}_N[X] \mod p$

$$(\mathbb{Z}/p\mathbb{Z})^N \simeq \mathbb{Z}_N[X] \mod p \simeq \frac{1}{p}\mathbb{Z}_N[X] \mod \mathbb{Z}$$

The plaintext space $\mathcal{M}$ is composed by exact multiples of $\frac{1}{p}$.



Plaintext addition $(\mu_1(X), \mu_2(X))$

$\mu_1(X) + \mu_2(X) := \mu_1(X) + \mu_2(X) \mod \mathbb{Z}.$

Plaintext product (Montgomery) $(\mu_1(X), \mu_2(X))$

$\mu_1(X) \boxtimes_p \mu_2(X) := p \cdot \tilde{\mu}_1(X) \cdot \tilde{\mu}_2(X) \mod \mathbb{Z}$, for lifts $\tilde{\mu}_1$ and $\tilde{\mu}_2$ in $\mathbb{R}_N[X]$

Problem of lift

Examples: $p = 3$, $\mu_1 = \frac{1}{3}$ and $\mu_2 = \frac{2}{3}$

- Exact product: $3(I_1 + \frac{1}{3})(I_2 + \frac{2}{3}) = I + \frac{2}{3} = +\frac{2}{3} \mod 1$, for all I_1, I_2 integers
- Product with noise and small element: $3 * 5.33333 * 10.66665 = 170.6662$
- Product with noise and big element: $3 * 12345678.33333 * 7654321.66665 = -.839\ldots$

- We need a small representative of the plaintext to keep the result correct.
- We should lift the ciphertext to small representative in $\mathbb{R}_N[X]$ (all coefficients in $[-1/2, 1/2)$).
- $\frac{1}{p} \gg noise$

Homomorphic operations

SANDBOXAQ ⊕inpher

Homomorphic addition $c_1 = (a_1, b_1)$, $c_2 = (a_2, b_2)$

$(a, b) = (a_1 + a_2, b_1 + b_2)$

Homomorphic product $c_1 = (a_1, b_1), c_2 = (a_2, b_2)$

$\mu_1 = b_1 - s \cdot a_1$ and $\mu_2 = b_2 - s \cdot a_2$

$$\begin{aligned} \mu_1 \boxtimes_p \mu_2 &= p(\tilde{b_1} - s \cdot \tilde{a}_1)(\tilde{b_2} - s \cdot \tilde{a}_2) \\ &= \underbrace{(p \cdot \tilde{b_1} \cdot \tilde{b_2})}_{C_0} - s \cdot \underbrace{(p \cdot \tilde{a}_1 \cdot \tilde{b_2} + p \cdot \tilde{a}_2 \cdot \tilde{b_1})}_{C_1} + s^2 \cdot \underbrace{(p \cdot \tilde{a}_1 \cdot \tilde{a}_2)}_{C_2} \\ &= (b - s \cdot a) \end{aligned}$$

The term s^2: relinearization with TRGSW encryption of s!

$$c_1 \boxtimes_p c_2 = (C_1, C_0) - TRGSW(s) \boxdot (C_2, 0)$$

The meaning of $a_1 a_2$, $b_1 b_2$ ...:

- small representatives in $\mathbb{R}_N[X]$

Fixed point

SANDBOXAQ ⊕inpher

Ciphertext

(a, b)

$$\mathbb{T}_N[X] + noise?$$

Circuits

$\mathcal{B} = (0, 1)$

Integers

$(\mathbb{Z}/p\mathbb{Z})^n$

Fixed point

$\mathbb{C}$

Fixed-point and Floating-points Numbers

SANDBOXAQ™ ⊕inpher

Floating point (float, double in C):

- $x = m.2^{\tau}$, with $m \in 2^{-\rho}.\mathbb{Z}$ and $\frac{1}{2} \leq |m| < 1$
- $\tau = \lceil log_2(x) \rceil$ data dependent and not public (not FHE-friendly)
- The exponent is always in sync with the data
  ex: $(1.23 \cdot 10^{-4}) * (7.24 \cdot 10^{-4}) = (8.90 \cdot 10^{-8})$

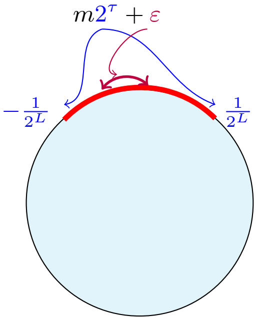
Fixed point:

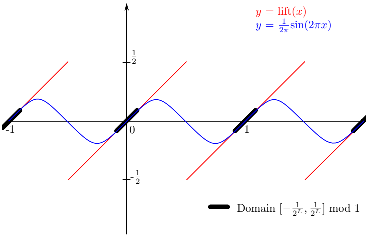
- $x = m.2^{\tau}$, with $m \in 2^{-\rho}.\mathbb{Z}$ and $0 \leq |m| < 1$,
- τ is public, thus FHE-friendly
- Risk of overflow (τ too small)
- Risk of underflow (τ too large)
  ex: $(0.000123 \cdot 10^{0}) * (0.000724 \cdot 10^{0}) = (0.000000 \cdot 10^{0})$

Addition is much tricker than you think!

- Given (m_1, τ_1), (m_2, τ_2), and τ.
- How do you compute $m.2^{\tau} = m_1.2^{\tau_1} + m_2.2^{\tau_2}$ with ρ bits of precision?
- Addition requires right shift and roundings, which are non-linear!

CKKS



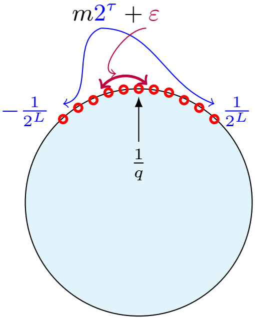


Continuous approach

- $x \times y = \text{Lift}(x) * \text{Lift}(y) \mod \mathbb{Z}$.
- ✔ This approach can preserve (or reduce) the interval $[-\frac{1}{2^L}, \frac{1}{2^L}]$
- ✔ Lift is a periodic function: approx by sinus (or other Fourier serie) wherever it matters...
- ✘ ...but sinus can only be approx by a polynomial, which recursively requires a product.

Fixed point: CKKS

SANDBOXAQ ⊕inpher



Discrete approach

- round a, b (and thus μ) on exact multiples of $\frac{1}{q}$ where $q \approx 2^{L+\rho}$.

✔ Brings us in the ring $\frac{1}{q}\mathbb{Z}_N[X] \mod \mathbb{Z}$

✔ Exact Montgomery product $q(b_1 - sa_1)(b_2 - sa_2)$

  - The meaning of $a_1 a_2$, $b_1 b_2$...: a_i, b_i are exact multiples of $\frac{1}{q}$

✘ Blows up the interval $[-\frac{1}{2^L}, \frac{1}{2^L}] \to [-\frac{1}{2^{L-\rho}}, \frac{1}{2^{L-\rho}}]$...
...works a leveled number of times.

Thank you for your attention!

Questions?

Appendix: Circuit Bootstrap mode versus Gate Bootstrap mode

SANDBOXAQ™ ⊕inpher

TFHE in Circuit Bootstrap mode
Bootstrap after many gates
(This work)

Input/Output

Plaintext (TLWE) → Ciphertext (TRGSW)
Bit Overhead: 262144

- Very fast : transition in $34\ \mu$s
- No so fast: circuit bootstrapped in 134 ms but after many gates
- Composition : LUT, (W)DFA

TFHE in Circuit bootstrap mode can evaluate LUT 16 to 8 in 1 sec

TFHE in Gate Bootstrap mode
Bootstrap between each gate
(TFHE 2016 + optimizations)

Input/Output

Plaintext (TLWE) → Ciphertext (TLWE)
Bit Overhead: 8000

- No so fast: bootstrapped binary gate runs in 13 ms
- All binary gates have the same cost
- Composition: unlimited

With TFHE we can compute 76 gates per second, for any circuit.

Appendix: Circuit Bootstrap mode versus Gate Bootstrap mode

TFHE in Circuit Bootstrap mode
Bootstrap after many gates
(This work)

Input/Output

Plaintext (TLWE) → Ciphertext (TRGSW)
Bit Overhead: 262144

- Very fast : transition in $34\ \mu$s
- No so fast: circuit bootstrapped in 134 ms but after many gates
- Composition : LUT, (W)DFA

TFHE in Circuit bootstrap mode can evaluate LUT 16 to 8 in 1 sec

TFHE in Gate Bootstrap mode
Bootstrap between each gate
(TFHE 2016 + optimizations)

Input/Output

Plaintext (TLWE) → Ciphertext (TLWE)
Bit Overhead: 8000

- No so fast: bootstrapped binary gate runs in 13 ms
- All binary gates have the same cost
- Composition: unlimited

With TFHE we can compute 76 gates per second, for any circuit.