Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds

I. Chillotti[1] N. Gama[2,1] M. Georgieva[3] M. Izabachène[4]

1 UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES université PARIS-SACLAY
2 inpher.io
3 gemalto
4 cea

Séminaire GTBAC Télécom ParisTech
April 6, 2017

Table of contents

Table of contents

Homomorphic Encryption

IDEA: perform computations on encrypted data, without decrypting it.

$b_1, b_2 \in \{0, 1\}$

$$\boxed{b_1} \qquad\qquad \boxed{b_1} \oplus_{hom} \boxed{b_2} \quad = \quad \boxed{b_1 \oplus b_2}$$

$$\longrightarrow$$

$$\boxed{b_2} \qquad\qquad \boxed{b_1} \wedge_{hom} \boxed{b_2} \quad = \quad \boxed{b_1 \wedge b_2}$$

Homomorphic Encryption

More generally

$$\boxed{b_1} \\ \vdots \\ \boxed{b_n} \quad \longrightarrow \quad \varphi_{hom}(\boxed{b_1}, \ldots, \boxed{b_n}) \quad = \quad \boxed{\varphi(b_1, \ldots, b_n)}$$

where $b_1, \ldots, b_n \in \{0, 1\}$ and φ is a boolean circuit.

Homomorphic Encryption

An **Homomorphic Encryption** scheme is composed by 4 algorithms:

Homomorphic Encryption

An **Homomorphic Encryption** scheme is composed by 4 algorithms:

- **Key Generation** KeyGen :

$$\lambda \longmapsto (sk, pk)$$

Homomorphic Encryption

An **Homomorphic Encryption** scheme is composed by 4 algorithms:

- **Key Generation** KeyGen :

$$\lambda \longmapsto (sk, pk)$$

- **Decryption** Dec (deterministic) :

$$(c, sk) \longmapsto m$$

Homomorphic Encryption

An **Homomorphic Encryption** scheme is composed by 4 algorithms:

- **Key Generation** KeyGen :

$$\lambda \longmapsto (sk, pk)$$

- **Decryption** Dec (deterministic) :

$$(c, sk) \longmapsto m$$

- **Encryption** Enc (randomized) :

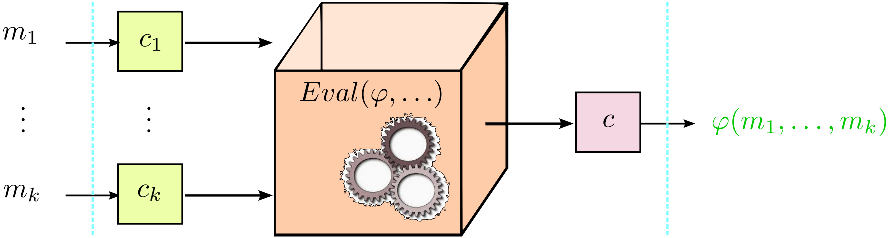
$$(m, pk) \longmapsto c$$

  such that $\mathsf{Dec}(c, sk) = m$

- **Evaluation** Eval (possibly randomized) :

$$(\varphi, c_1, \ldots, c_k) \longmapsto c$$

such that $\mathsf{Dec}(c, sk) = \varphi(m_1, \ldots, m_k)$

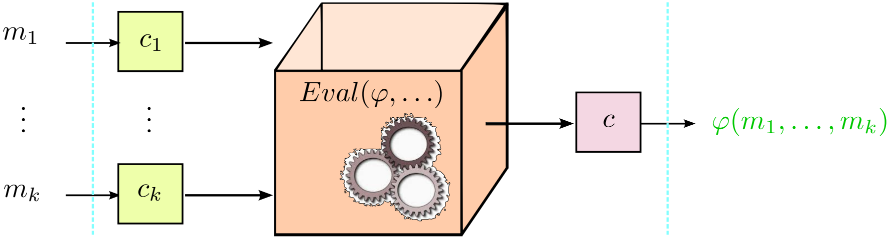


Homomorphic Encryption

- **Evaluation** Eval (possibly randomized) :

$$(\varphi, c_1, \ldots, c_k) \longmapsto c$$

such that $\mathsf{Dec}(c, sk) = \varphi(m_1, \ldots, m_k)$



A scheme that can homomorphically evaluate all functions/circuits is said **Fully Homomorphic** (FHE).

Applications

Applications

Statistic computations on sensitive data

Applications

Statistic computations on sensitive data

Secure multiparty computation

Applications

Statistic computations on sensitive data

Secure multiparty computation

Electronic voting

Applications

Statistic computations on sensitive data

Secure multiparty computation

Electronic voting

Cloud computing

Applications

Statistic computations on sensitive data

Secure multiparty computation

Electronic voting

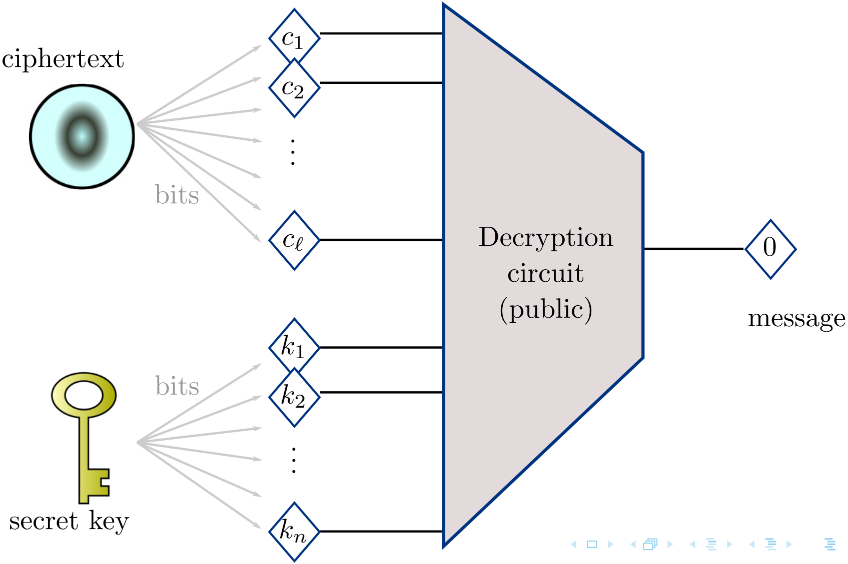
Cloud computing

and even more...

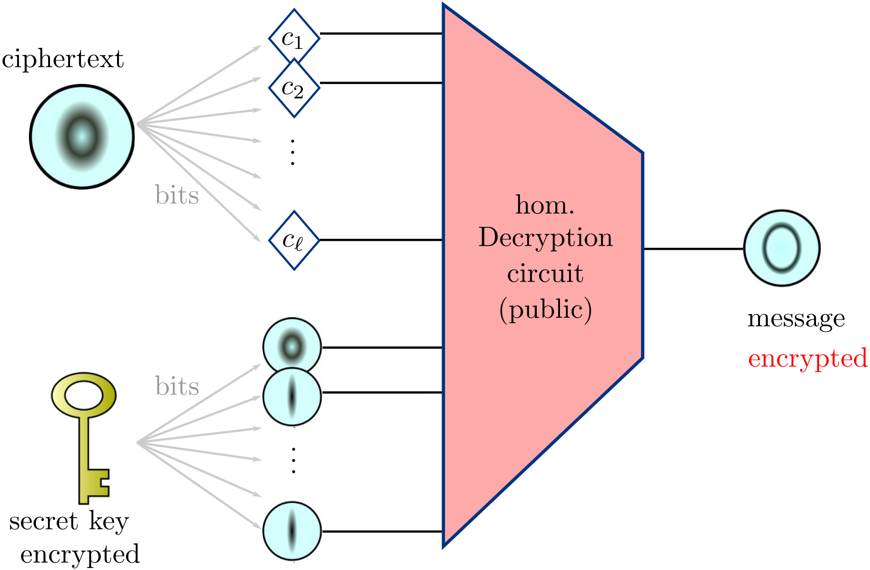
A world full of noise...

anim.html

Bootstrapping now



Bootstrapping now



Bootstrapping now

Bootstrapping is the most expensive part of the entire homomorphic procedure

- Original idea by Gentry [Gen09]
- Last years: work to reduce the execution time and memory consuming

...but a lot have to be done!

Table of contents

LWE

- LWE = Learning With Errors [Reg05]
- Ring-LWE [LPR10]

LWE

- LWE = Learning With Errors [Reg05]
- Ring-LWE [LPR10]

In our paper

- LWE: definition similar to [BLPRS13],[CS15],[CGGI16]
- TLWE: generalized definition similar to [BGV12]

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z} = \mathbb{R} \mod 1$

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

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z} = \mathbb{R} \mod 1$

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

✔ It is a group: $x + y \mod 1$ and $-x \mod 1$

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z} = \mathbb{R} \mod 1$

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

- ✔ It is a group: $x + y \mod 1$ and $-x \mod 1$
- ✔ It is a $\mathbb{Z}$-module: $0 \cdot \frac{1}{2} = 0$ is defined!

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z} = \mathbb{R} \mod 1$

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

- ✔ It is a group: $x + y \mod 1$ and $-x \mod 1$
- ✔ 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!

The real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z} = \mathbb{R} \mod 1$

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

- ✔ It is a group: $x + y \mod 1$ and $-x \mod 1$
- ✔ 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!

Vectors/matrices

By extension, $(\mathbb{T}^n, +, .)$ is a $\mathbb{Z}$-module

- $\begin{bmatrix} 3 & -2 & 4 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & -2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.252 & 0.672 \\ 0.231 & 0.991 \end{bmatrix} \right)$
- $= \left(\begin{bmatrix} 3 & -2 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & -2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix} \right) \cdot \begin{bmatrix} 0.252 & 0.672 \\ 0.231 & 0.991 \end{bmatrix}$

Torus polynomials $\mathbb{T}_N[X]$

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

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

Torus polynomials $\mathbb{T}_N[X]$

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

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

Examples

- $(1 + 2X) \cdot (\frac{1}{3} + \frac{4}{7}X) =$

Torus polynomials $\mathbb{T}_N[X]$

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

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

Examples

- $(1 + 2X) \cdot (\frac{1}{3} + \frac{4}{7}X) = (\frac{4}{21} + \frac{5}{21}X) \mod (X^2 + 1) \mod 1$

Torus polynomials $\mathbb{T}_N[X]$

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

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

Examples

- $(1 + 2X) \cdot (\frac{1}{3} + \frac{4}{7}X) = (\frac{4}{21} + \frac{5}{21}X) \mod (X^2 + 1) \mod 1$
- Decompose $(\frac{3}{8} + \frac{7}{8}X)$ over $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}]$ with **small** coefs

Torus polynomials $\mathbb{T}_N[X]$

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

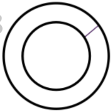
- Here, $\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$
- And $\mathbb{T}_N[X] = \mathbb{T}[X] \mod (X^N + 1)$

Examples

- $(1 + 2X) \cdot (\frac{1}{3} + \frac{4}{7}X) = (\frac{4}{21} + \frac{5}{21}X) \mod (X^2 + 1) \mod 1$
- Decompose $(\frac{3}{8} + \frac{7}{8}X)$ over $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}]$ with **small** coefs
  $(\frac{3}{8} + \frac{7}{8}X) = (0 + X) \cdot \frac{1}{2} + (1 + X) \cdot \frac{1}{4} + (1 + X) \cdot \frac{1}{8}$

LWE symmetric encryption

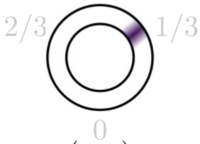
LWE symmetric encryption



Example: $\mathcal{M} = \{0, 1/3, 2/3\} \; mod \; 1$

$\mu = 1/3 \; mod \; 1 \in \mathcal{M}$

LWE symmetric encryption



$(\ , \varphi)$

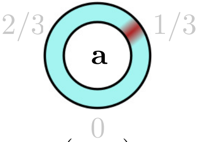
Example: $\mathcal{M} = \{0, 1/3, 2/3\} \ mod \ 1$

$\mu = 1/3 \ mod \ 1 \in \mathcal{M}$

LWE Encryption

1. Choose $\varphi = \mu + \mathit{Gaussian\ Error}$

LWE symmetric encryption



$(\mathbf{a}, \varphi)$

Example: $\mathcal{M} = \{0, 1/3, 2/3\}$ mod 1

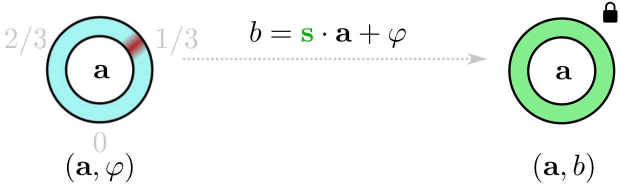
$\mu = 1/3$ mod $1 \in \mathcal{M}$

LWE Encryption

1. Choose $\varphi = \mu + \textit{Gaussian Error}$
2. Choose a random mask $\mathbf{a} \in \mathbb{T}^n$

LWE symmetric encryption

secret key: $\mathbf{s} \in \{0,1\}^n$



Example: $\mathcal{M} = \{0, 1/3, 2/3\}$ *mod* 1

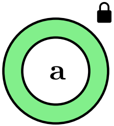
$\mu = 1/3$ *mod* $1 \in \mathcal{M}$

LWE Encryption

1. Choose $\varphi = \mu + $ *Gaussian Error*
2. Choose a random mask $\mathbf{a} \in \mathbb{T}^n$
3. Return the locked representation $(\mathbf{a}, b)$

LWE symmetric encryption

secret key: $\mathbf{s} \in \{0,1\}^n$

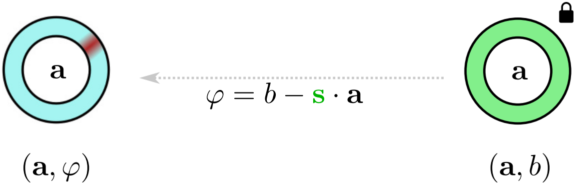


$(\mathbf{a}, b)$

LWE Decryption

LWE symmetric encryption

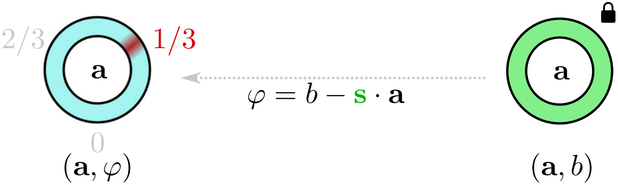
secret key: $\mathbf{s} \in \{0,1\}^n$



LWE Decryption

1. Unlock the representation $(\mathbf{a}, \varphi)$

secret key: $\mathbf{s} \in \{0,1\}^n$

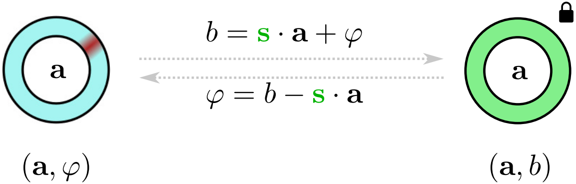


LWE Decryption

1. Unlock the representation $(\mathbf{a}, \varphi)$
2. Round φ to the nearest message $\mu \in \mathcal{M}$

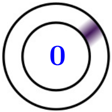
LWE symmetric encryption

secret key: $\mathbf{s} \in \{0,1\}^n$



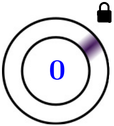
LWE symmetric encryption

secret key: $\mathbf{s} \in \{0,1\}^n$



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

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



$(\mathbf{a}, b)$

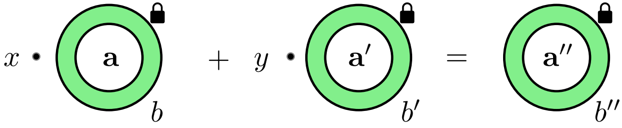
Trivial LWE samples

- LWE samples with mask $\mathbf{a} = \mathbf{0}$ are trivial.
- They never occur in general

...but are still worth mentionning!

LWE

Homomorphic Properties

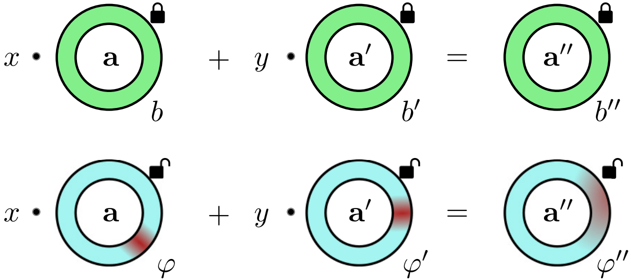


$$\mathbf{a}'' = x \cdot \mathbf{a} + y \cdot \mathbf{a}'$$

$$b'' = x \cdot b + y \cdot b'$$

LWE

Homomorphic Properties

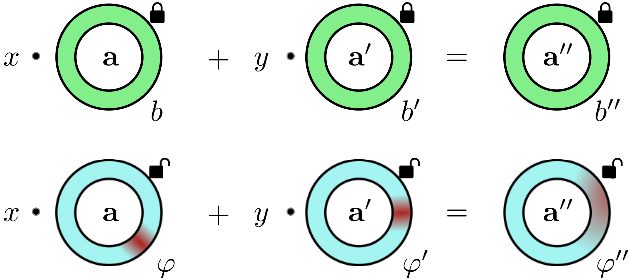


$$\mathbf{a}'' = x \cdot \mathbf{a} + y \cdot \mathbf{a}'$$

$$b'' = x \cdot b + y \cdot b'$$

$$\varphi'' = x \cdot \varphi + y \cdot \varphi'$$

Homomorphic Properties



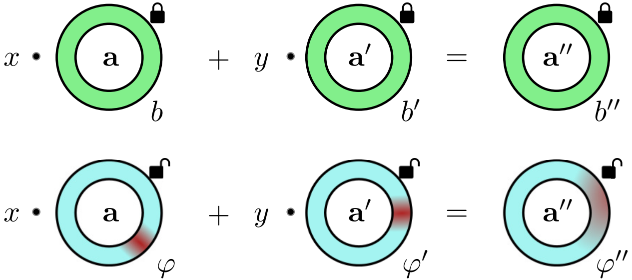
$$\mathbf{a}'' = x \cdot \mathbf{a} + y \cdot \mathbf{a}'$$

$$b'' = x \cdot b + y \cdot b'$$

$$\varphi'' = x \cdot \varphi + y \cdot \varphi'$$

$\mu = \mathbb{E}(\varphi)$ $\qquad \mu'$ $\qquad \mu''$ $\qquad \mu'' = x \cdot \mu + y \cdot \mu'$

Homomorphic Properties



$$\mathbf{a}'' = x \cdot \mathbf{a} + y \cdot \mathbf{a}'$$

$$b'' = x \cdot b + y \cdot b'$$

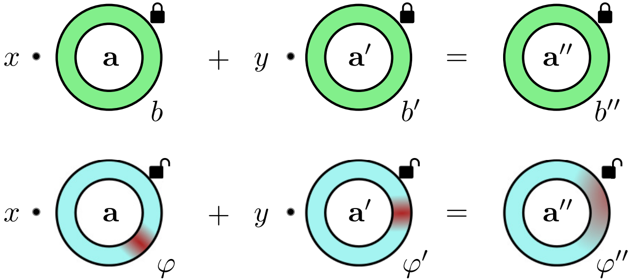
$$\varphi'' = x \cdot \varphi + y \cdot \varphi'$$

$\mu = \mathbb{E}(\varphi)$ $\quad \mu'$ $\quad \mu''$ $\quad \mu'' = x \cdot \mu + y \cdot \mu'$

$\alpha = \mathrm{stdev}(\varphi)$ $\quad \alpha'$ $\quad \alpha''$ $\quad \alpha''^2 = x^2\alpha^2 + y^2\alpha'$

LWE

Homomorphic Properties



$\mathbf{a}'' = x \cdot \mathbf{a} + y \cdot \mathbf{a}'$

$b'' = x \cdot b + y \cdot b'$

$\varphi'' = x \cdot \varphi + y \cdot \varphi'$

$\mu = \mathbb{E}(\varphi)$ μ' μ'' $\mu'' = x \cdot \mu + y \cdot \mu'$

$\alpha = \text{stdev}(\varphi)$ α' α'' $\alpha''^2 = x^2\alpha^2 + y^2\alpha'$

Ω: *The only proba. space where this intuitive picture makes sense!*

LWE

- LWE = Learning With Errors [Reg05]
- Ring-LWE [LPR10]

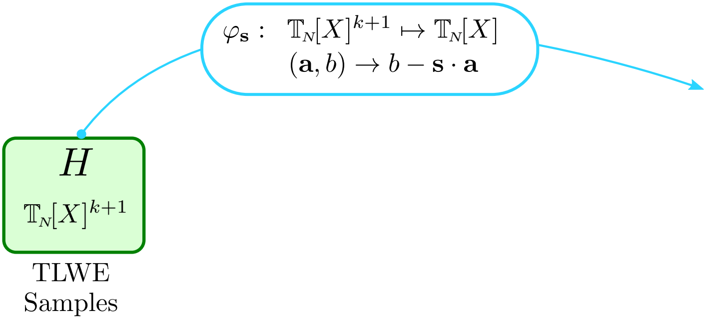
LWE

- LWE = Learning With Errors [Reg05]
- Ring-LWE [LPR10]

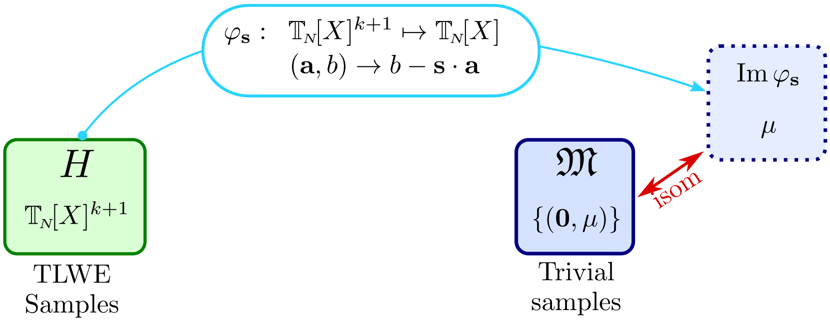
In our paper

- LWE: definition similar to [BLPRS13],[CS15],[CGGI16]
- TLWE: generalized definition similar to [BGV12]

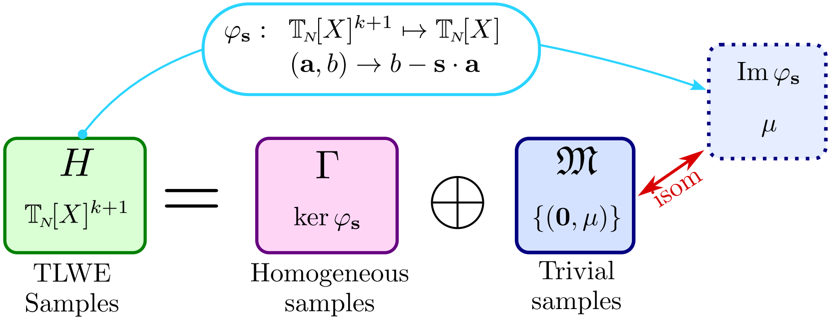
TLWE Encryption



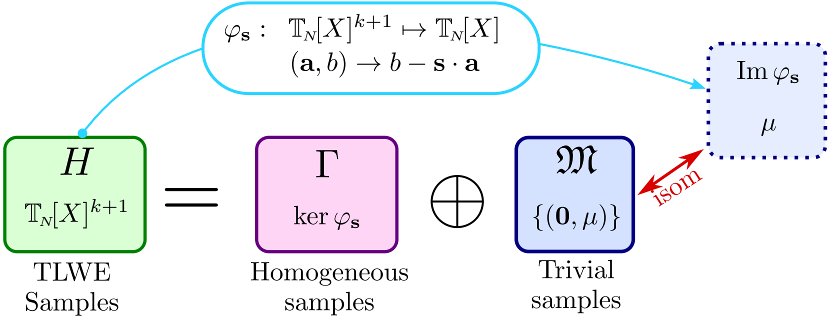
TLWE Encryption

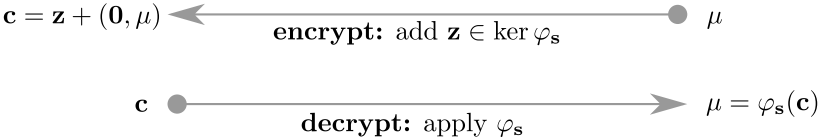


TLWE Encryption

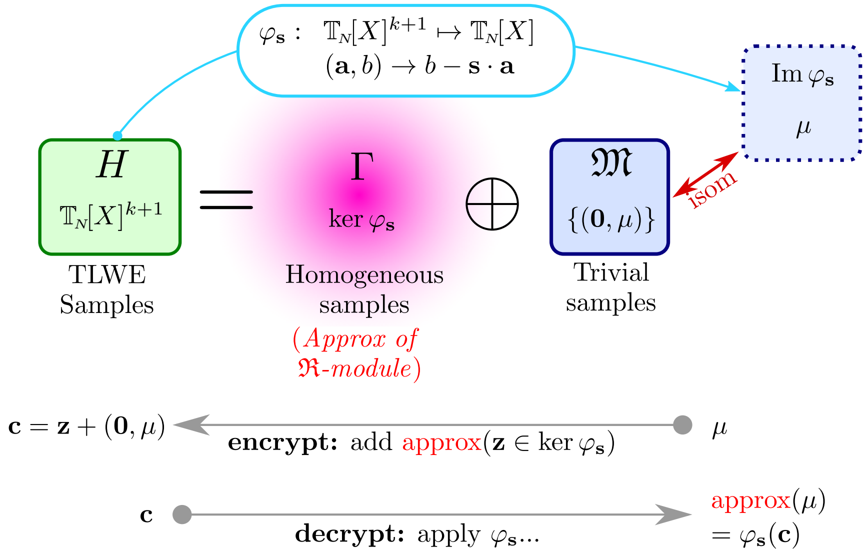


TLWE Encryption





TLWE Encryption



TLWE Encryption

n $\varphi_{\mathbf{s}}$

μ

How to recover μ **exactly**?

Option 1: $\mu = \mathbb{E}(\varphi_{\mathbf{s}}(\mathbf{c}))$

(in the relevant proba. space)

The Ω-space logic

Option 2: $\mu = \text{round}(\varphi_{\mathbf{s}}(\mathbf{c}))$

On a given finite message space $\mathcal{M}$

The logic of the decryption algorithm

$\mathbf{c} =$

$\mathbf{c}$ → **decrypt:** apply $\varphi_{\mathbf{s}}$...

approx(μ)

$= \varphi_{\mathbf{s}}(\mathbf{c})$

Table of contents

We want FHE!

What is still missing to have **Fully** Homomorphic Encryption?

We want FHE!

What is still missing to have **Fully** Homomorphic Encryption?

- GSW [GSW13] is a FHE scheme based on LWE
- Relies on a gadget decomposition function

GSW

We want FHE!

What is still missing to have **Fully** Homomorphic Encryption?

- GSW [GSW13] is a FHE scheme based on LWE
- Relies on a gadget decomposition function

In this talk

- Abstraction of [GSW13] by [GINX16]
- TGSW: "GSW" on $\mathbb{T}$

The gadget

$$\mathbf{v} = \left(\; v_1 \; | \ldots | \; v_{k+1} \; \right) \in H$$

$$\mathbf{h} = \left(\begin{array}{c|c|c} 1/2 & \ldots & 0 \\ 1/2^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 1/2^\ell & \ldots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \ldots & 1/2 \\ 0 & \ldots & 1/2^2 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 1/2^\ell \end{array} \right)$$

h generating family of H

$\mathbf{h} \in \mathcal{M}_{\ell',k+1}(\mathbb{T}_N[X])$

- $\mathbf{h}$ is block diagonal super-increasing
- We are able to decompose elements in the sub-module H
- The coefficients in the decomposition are small
- Approximated decomposition (up to some precision parameters)
- Improve time and memory requirements for a small amount of additional noise

TGSW

Parameters

- Let $H = \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$
- $\mathbf{h} = (h_1, \ldots, h_l) \in H^{\ell'}$ a super-increasing generating family of H
- $Dec_{\mathbf{h}}$ the "small" decomposition function from $H \to \mathfrak{R}^{\ell'}$ ($\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$) such that

$$Dec_{\mathbf{h}}(x) \cdot \mathbf{h} = x \text{ for all } x \in H$$

- $\Gamma = \ker_{\varphi_{\mathbf{s}}}$ denotes homogeneous TLWE samples

TGSW

Parameters

- Let $H = \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$
- $\mathbf{h} = (h_1, \ldots, h_l) \in H^{\ell'}$ a super-increasing generating family of H
- $Dec_{\mathbf{h}}$ the "small" decomposition function from $H \to \mathfrak{R}^{\ell'}$ ($\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$) such that

$$Dec_{\mathbf{h}}(x) \cdot \mathbf{h} = x \text{ for all } x \in H$$

- $\Gamma = \ker_{\varphi_{\mathbf{s}}}$ denotes homogeneous TLWE samples

Encryption:

$C = Z + \mu \cdot \mathbf{h}$ where $Z \in \Gamma^{\ell'}$

TGSW

Parameters

- Let $H = \mathbb{T}_N[X]^k \times \mathbb{T}_N[X]$
- $\mathbf{h} = (h_1, \dots, h_l) \in H^{\ell'}$ a super-increasing generating family of H
- $Dec_{\mathbf{h}}$ the "small" decomposition function from $H \to \mathfrak{R}^{\ell'}$ ($\mathfrak{R} = \mathbb{Z}[X]/(X^N + 1)$) such that

$$Dec_{\mathbf{h}}(x) \cdot \mathbf{h} = x \text{ for all } x \in H$$

- $\Gamma = \ker_{\varphi_{\mathbf{s}}}$ denotes homogeneous TLWE samples

Encryption:

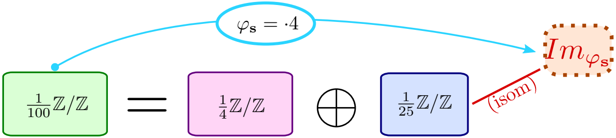
$C = Z + \mu \cdot \mathbf{h}$ where $Z \in \Gamma^{\ell'}$

Homomorphic operations:

Let $C_1 = Z_1 + \mu_1 \cdot \mathbf{h}$ and $C_2 = Z_2 + \mu_2 \cdot \mathbf{h}$

- **Linear combinations:** $\delta_1 C_1 + \delta_2 C_2$ encrypts $\delta_1\mu_1 + \delta_2\mu_2$ ($\delta_i \in \mathfrak{R}$)
- **Multiplication :** $Dec_{\mathbf{h}}(C_1) \cdot C_2$ encrypts $\mu_1\mu_2$

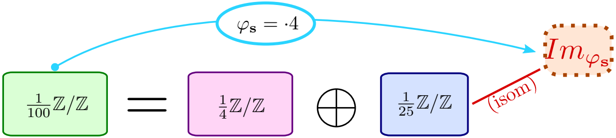
Toy example (without noise)



Parameters

- $H = \frac{1}{100}\mathbb{Z}/\mathbb{Z} = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \oplus \frac{1}{25}\mathbb{Z}/\mathbb{Z}$ (is a $\mathbb{Z}$-module)
- $\mathbf{h} = \left(\frac{1}{100}, \frac{2}{100}, \frac{5}{100}, \frac{10}{100}, \frac{20}{100}, \frac{50}{100}\right)$
- $Dec_{\mathbf{h}}$: decomposition in Euro coins
- $\Gamma = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \subset H$: modulo of the code

Toy example (without noise)



Parameters

- $H = \frac{1}{100}\mathbb{Z}/\mathbb{Z} = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \oplus \frac{1}{25}\mathbb{Z}/\mathbb{Z}$ (is a $\mathbb{Z}$-module)
- $\mathbf{h} = \left(\frac{1}{100}, \frac{2}{100}, \frac{5}{100}, \frac{10}{100}, \frac{20}{100}, \frac{50}{100}\right)$
- $Dec_{\mathbf{h}}$: decomposition in Euro coins
- $\Gamma = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \subset H$: modulo of the code

Samples

$$C_1 = \left(\frac{32}{100}, \frac{14}{100}, \frac{60}{100}, \frac{45}{100}, \frac{90}{100}, \frac{0}{100}\right) = \left(\frac{1}{4}, \frac{0}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}\right) + 7 \cdot \mathbf{h}$$

$$C_2 = \left(\frac{73}{100}, \frac{21}{100}, \frac{40}{100}, \frac{5}{100}, \frac{35}{100}, \frac{50}{100}\right) = \left(\frac{3}{4}, \frac{1}{4}, \frac{2}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}\right) - 2 \cdot \mathbf{h}$$

Toy example (without noise)

Multiplication:

$$Dec_{\mathbf{h}}(C_1) \cdot C_2 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 73/100 \\ 21/100 \\ 40/100 \\ 5/100 \\ 35/100 \\ 50/100 \end{bmatrix}$$

$$= \left(\frac{61}{100}, \frac{47}{100}, \frac{55}{100}, \frac{10}{100}, \frac{20}{100}, \frac{0}{100} \right)$$

Verification: does encode $7 \cdot (-2) = 11 \mod 25$

$$\left(\frac{61}{100}, \frac{47}{100}, \frac{55}{100}, \frac{10}{100}, \frac{20}{100}, \frac{0}{100} \right) = \left(\frac{2}{4}, \frac{1}{4}, \frac{0}{4}, \frac{0}{4}, \frac{0}{4}, \frac{2}{4} \right) + 11 \cdot \mathbf{h}$$

Toy example (without noise)

Multiplication:

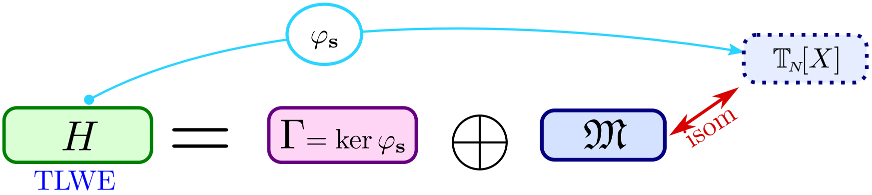
$$Dec_{\mathbf{h}}(C_1) \cdot C_2 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 73/100 \\ 21/100 \\ 40/100 \\ 5/100 \\ 35/100 \\ 50/100 \end{bmatrix}$$

$$= \left(\frac{61}{100}, \frac{47}{100}, \frac{55}{100}, \frac{10}{100}, \frac{20}{100}, \frac{0}{100} \right)$$

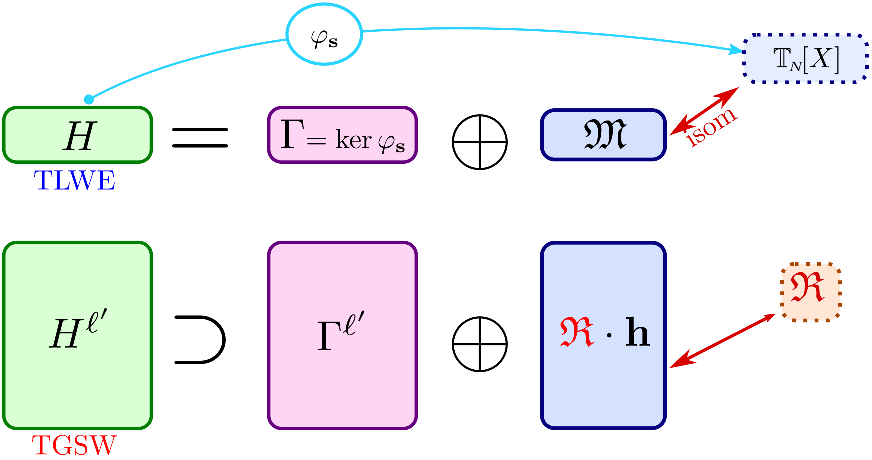
Verification: does encode $7 \cdot (-2) = 11 \mod 25$

$$\left(\frac{61}{100}, \frac{47}{100}, \frac{55}{100}, \frac{10}{100}, \frac{20}{100}, \frac{0}{100} \right) = \left(\frac{2}{4}, \frac{1}{4}, \frac{0}{4}, \frac{0}{4}, \frac{0}{4}, \frac{2}{4} \right) + 11 \cdot \mathbf{h}$$

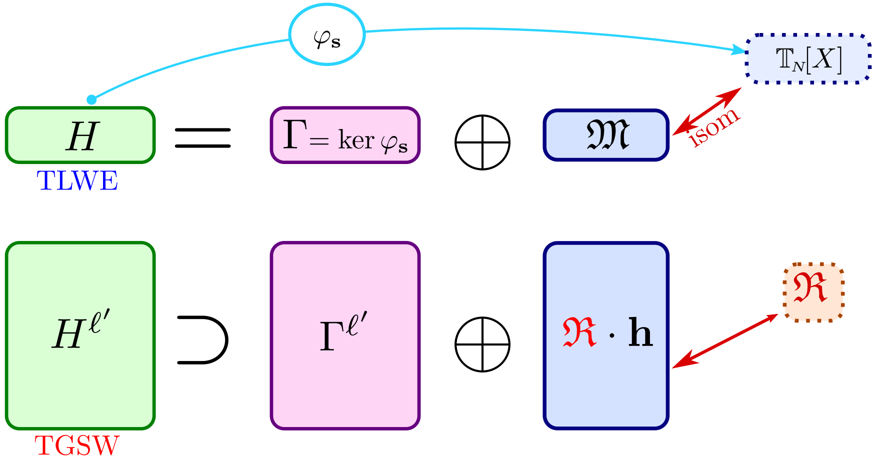
TLWE and TGSW



TLWE and TGSW



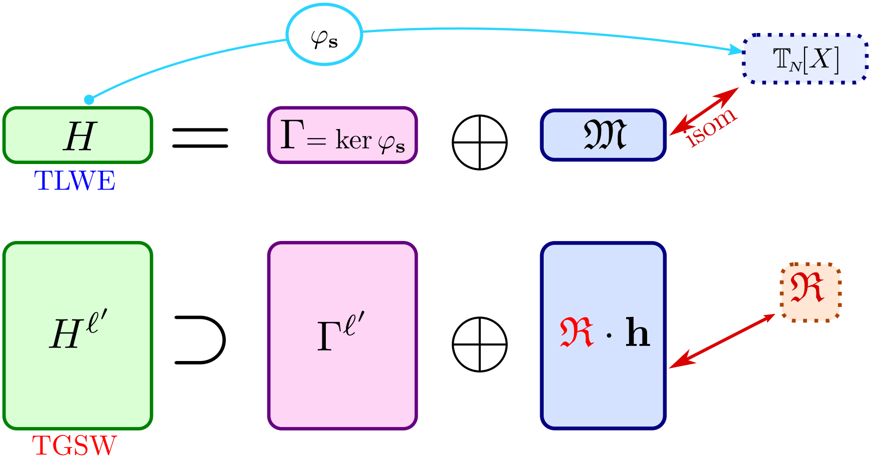
TLWE and TGSW



$\forall \mathbf{e} \in \mathfrak{R}^{\ell'}, \forall A \in \mathfrak{R}, \forall b \in \mathbb{T}_N[X]$:

$\mathbf{e} \cdot \mathtt{TGSW}(A)$ is a $\mathtt{TLWE}$ of $A \cdot \varphi_{\mathbf{s}}(\mathbf{e} \cdot \mathbf{h})$

TLWE and TGSW



$\forall \mathbf{e} \in \mathfrak{R}^{\ell'}, \forall A \in \mathfrak{R}, \forall b \in \mathbb{T}_N[X]$:

$\mathbf{e} \cdot \texttt{TGSW}(A)$ is a $\texttt{TLWE}$ of $A \cdot \varphi_{\mathbf{s}}(\mathbf{e} \cdot \mathbf{h})$

$\implies \mathrm{Decomp}_{\mathbf{h}}(\texttt{TLWE}(b)) \cdot \texttt{TGSW}(A)$ is a $\texttt{TLWE}$ of $A \cdot b$

Toy example (WITH noise)

Parameters

- $H = \frac{1}{100}\mathbb{Z}/\mathbb{Z} = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \oplus \frac{1}{25}\mathbb{Z}/\mathbb{Z}$ (is a $\mathbb{Z}$-module)
- $\mathbf{h} = \left(\frac{1}{100}, \frac{2}{100}, \frac{5}{100}, \frac{10}{100}, \frac{20}{100}, \frac{50}{100}\right)$
- $Dec_{\mathbf{h}}$: decomposition in Euro coins
- $\Gamma = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \subset H$: modulo of the code

Samples

$$C_1 = \left(\frac{31}{100}, \frac{16}{100}, \frac{63}{100}, \frac{46}{100}, \frac{89}{100}, \frac{0}{100}\right)$$

$$= \left[\left(\frac{1}{4}, \frac{0}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}\right) + \left(-\frac{1}{100}, \frac{2}{100}, \frac{3}{100}, \frac{1}{100}, -\frac{1}{100}, \frac{1}{100}\right)\right] + 7 \cdot \mathbf{h}$$

$$C_2 = \left(\frac{71}{100}, \frac{23}{100}, \frac{37}{100}, \frac{5}{100}, \frac{33}{100}, \frac{48}{100}\right)$$

$$= \left[\left(\frac{3}{4}, \frac{1}{4}, \frac{2}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}\right) + \left(-\frac{2}{100}, \frac{2}{100}, -\frac{3}{100}, \frac{0}{100}, -\frac{2}{100}, -\frac{2}{100}\right)\right] - 2 \cdot \mathbf{h}$$

Toy example (WITH noise)

Parameters

- $H = \frac{1}{100}\mathbb{Z}/\mathbb{Z} = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \oplus \frac{1}{25}\mathbb{Z}/\mathbb{Z}$ (is a $\mathbb{Z}$-module)
- $\mathbf{h} = \left(\frac{1}{100}, \frac{2}{100}, \frac{5}{100}, \frac{10}{100}, \frac{20}{100}, \frac{50}{100}\right)$
- $Dec_{\mathbf{h}}$: decomposition in Euro coins
- $\Gamma = \frac{1}{4}\mathbb{Z}/\mathbb{Z} \subset H$: modulo of the code

Samples

$$C_1 = \left(\frac{31}{100}, \frac{16}{100}, \frac{63}{100}, \frac{46}{100}, \frac{89}{100}, \frac{0}{100}\right)$$

$$= \left[\left(\frac{1}{4}, \frac{0}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}\right) + \left(-\frac{1}{100}, \frac{2}{100}, \frac{3}{100}, \frac{1}{100}, -\frac{1}{100}, \frac{1}{100}\right)\right] + 7 \cdot \mathbf{h}$$

$$C_2 = \left(\frac{71}{100}, \frac{23}{100}, \frac{37}{100}, \frac{5}{100}, \frac{33}{100}, \frac{48}{100}\right)$$

$$= \left[\left(\frac{3}{4}, \frac{1}{4}, \frac{2}{4}, \frac{1}{4}, \frac{3}{4}, \frac{2}{4}\right) + \left(-\frac{2}{100}, \frac{2}{100}, -\frac{3}{100}, \frac{0}{100}, -\frac{2}{100}, -\frac{2}{100}\right)\right] - 2 \cdot \mathbf{h}$$

Toy example (WITH noise)

Multiplication:

$$Dec_{\mathbf{h}}(C_{1,1}) \cdot C_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 71/100 \\ 23/100 \\ 37/100 \\ 5/100 \\ 33/100 \\ 48/100 \end{bmatrix}$$

$$Dec_{\mathbf{h}}(C_{1,1}) \cdot C_2 = \left(\frac{9}{100} \right)$$

Verification: does encode $7 \cdot (-2) = 11 \mod 25$

$$\left(\frac{9}{100} \right) = \left[\left(\frac{0}{4} \right) - \left(\frac{2}{100} \right) \right] + 11 \cdot h_1$$

Product

External product (found independently by [BP16])

$$\boxdot : TGSW \times TLWE \longrightarrow TLWE$$
$$(A, \mathbf{b}) \longmapsto A \boxdot \mathbf{b} = Dec_{\mathbf{h},\beta,\epsilon}(\mathbf{b}) \cdot A$$
$$(\mu_A, \mu_{\mathbf{b}}) \longmapsto \mu_A \cdot \mu_{\mathbf{b}}$$

where $Dec_{\mathbf{h},\beta,\epsilon}$ is the approximate gadget decomposition

Product

External product (found independently by [BP16])

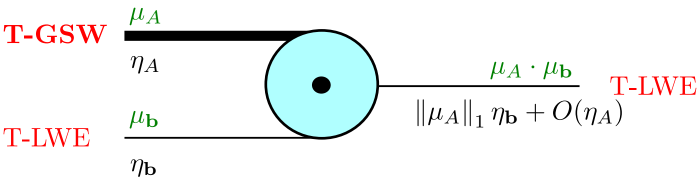
$$\begin{aligned}\boxdot\colon TGSW \times TLWE &\longrightarrow TLWE\\ (A, \mathbf{b}) &\longmapsto A \boxdot \mathbf{b} = Dec_{\mathbf{h},\beta,\epsilon}(\mathbf{b}) \cdot A\\ (\mu_A, \mu_{\mathbf{b}}) &\longmapsto \mu_A \cdot \mu_{\mathbf{b}}\end{aligned}$$

where $Dec_{\mathbf{h},\beta,\epsilon}$ is the approximate gadget decomposition

Internal product (classical)

$$\begin{aligned}\boxtimes\colon TGSW \times TGSW &\longrightarrow TGSW\\ (A, B) &\longmapsto A \boxtimes B = \begin{bmatrix} A \boxdot \mathbf{b_1} \\ \vdots \\ A \boxdot \mathbf{b_{(k+1)\ell}} \end{bmatrix}\\ (\mu_A, \mu_B) &\longmapsto \mu_A \cdot \mu_B\end{aligned}$$

Product



$$\|\mathsf{Err}(A \boxdot \mathbf{b})\|_\infty \leq \boxed{\ell' N \beta \eta_A + \|\mu_A\|_1 (1 + kN)\epsilon} + \boxed{\|\mu_A\|_1 \eta_{\mathbf{b}}}$$

where β and ϵ are the parameters used in the decomposition $Dec_{\mathbf{h},\beta,\epsilon}(\mathbf{b})$.

Table of contents

Faster bootstrapping

We applied our result to the fast bootstrapping proposed by Ducas and Micciancio (Eurocrypt 2015)

[DM15]: homomorphic NAND gate with fast bootstrapping in ~ 0.69 seconds

Faster bootstrapping

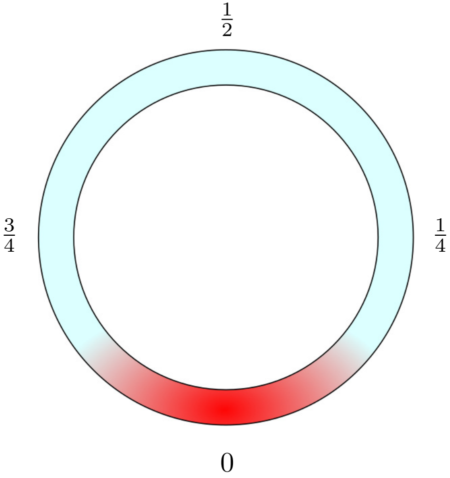
We applied our result to the fast bootstrapping proposed by Ducas and Micciancio (Eurocrypt 2015)

[DM15]: homomorphic NAND gate with fast bootstrapping in ~ 0.69 seconds

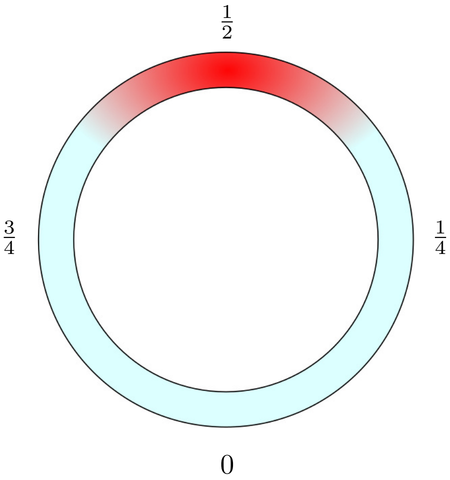
We replaced all the internal products in the bootstrapping procedure with the external one.

Result: (with further optimizations) we had a speed-up of a factor ~ 12 (bootstrapping in ~ 0.052 seconds)

Bootstrapping

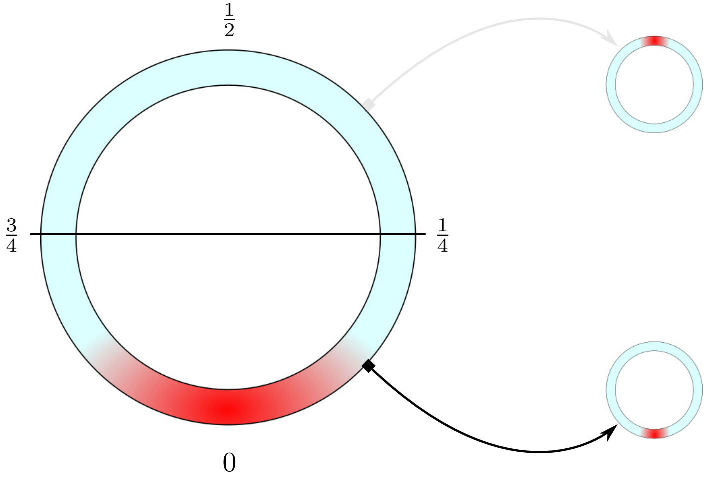


Bootstrapping



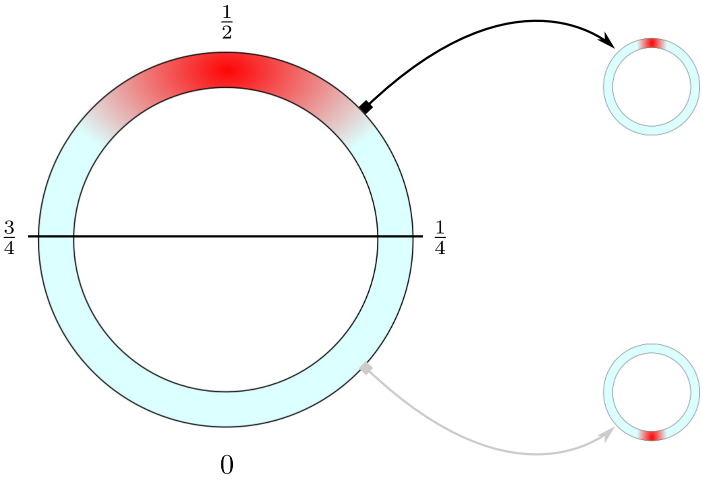
Bootstrapping

[Gentry09]-style bootstrap



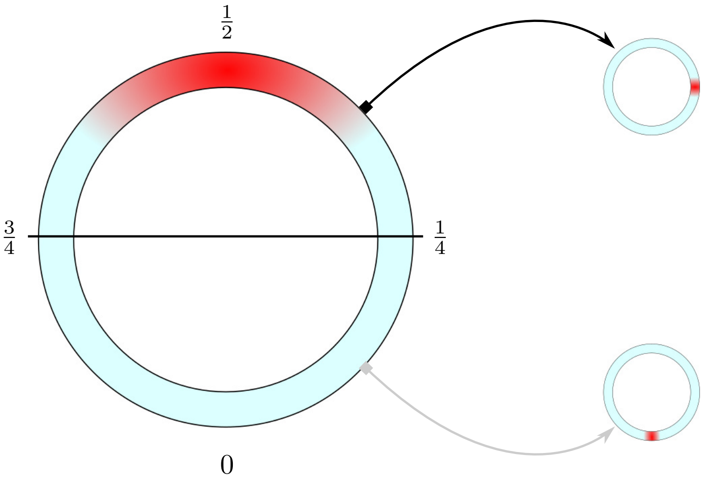
Bootstrapping

[Gentry09]-style bootstrap



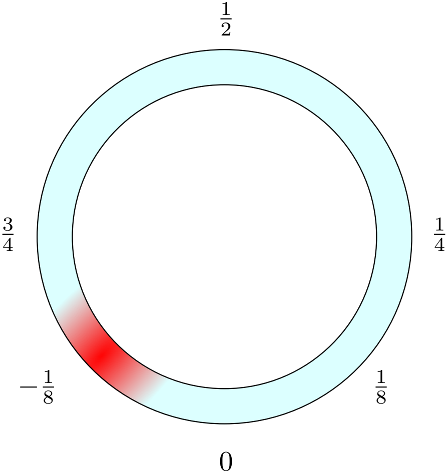
Bootstrapping

[DM15]-style bootstrap



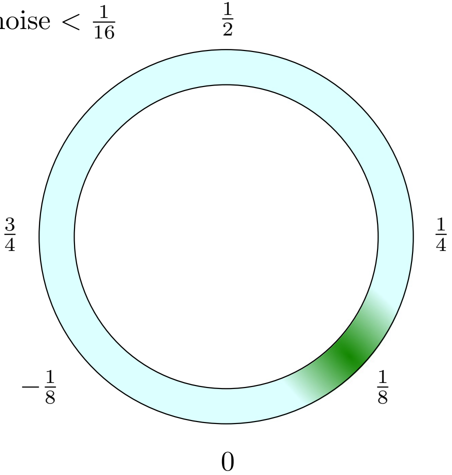
Gate Bootstrapping

false $:= \mathrm{LWE}(-\frac{1}{8})$, noise$< \frac{1}{16}$

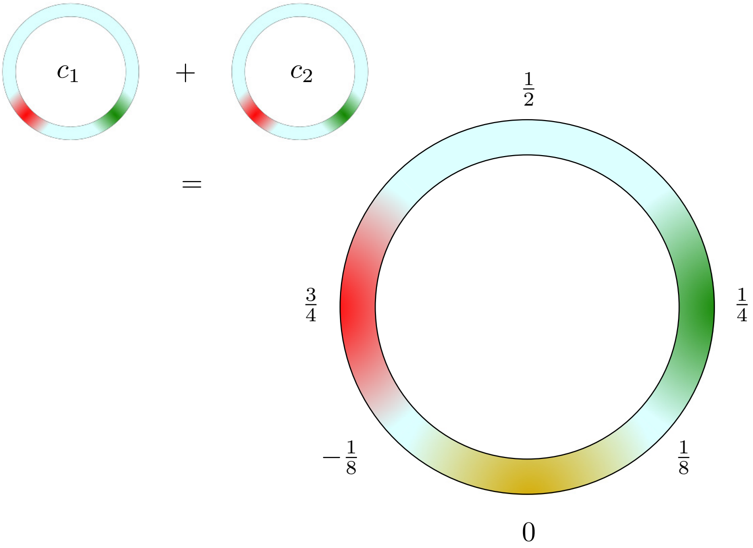


Gate Bootstrapping

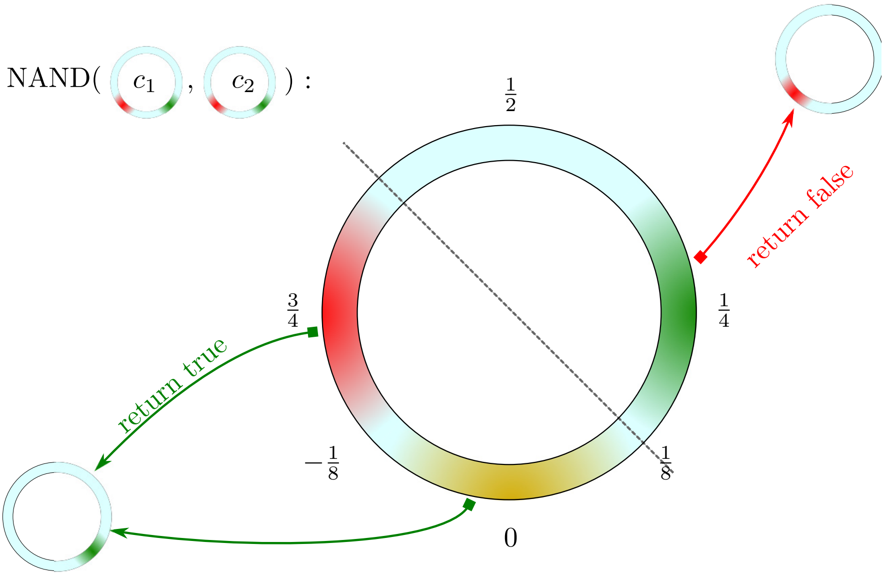
$\textcolor{green}{\text{true}} := \text{LWE}(+\frac{1}{8})$, noise $< \frac{1}{16}$



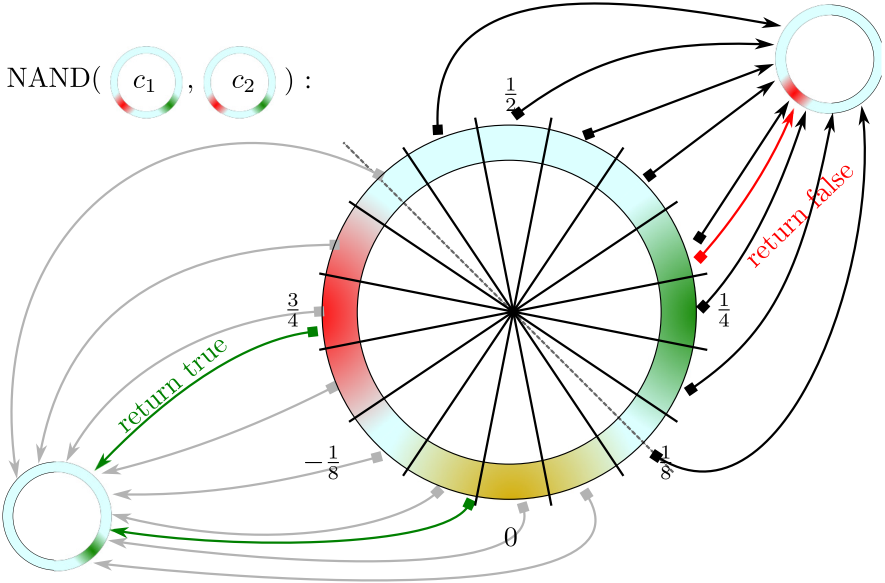
Gate Bootstrapping



Gate Bootstrapping

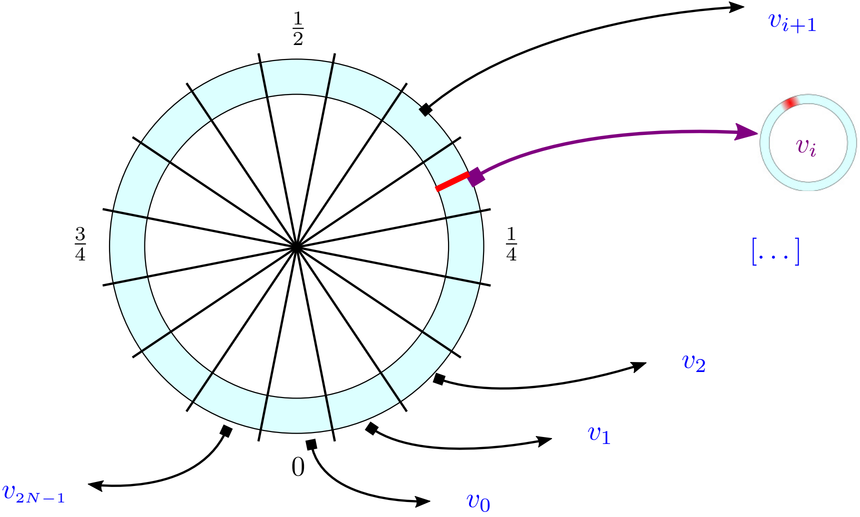


Gate Bootstrapping



Gate Bootstrapping

[DM15/BR15]-(revisited)



Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

How to rotate by $-\varphi_{\mathbf{s}}(\mathbf{a}, b) = -b + \sum_{i=1}^{n} a_i s_i$?

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

How to rotate by $-\varphi_{\mathbf{s}}(\mathbf{a}, b) = -b + \sum_{i=1}^{n} a_i s_i$?

1. Multiply by X^{-b}

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

How to rotate by $-\varphi_{\mathbf{s}}(\mathbf{a}, b) = -b + \sum_{i=1}^{n} a_i s_i$?

1. Multiply by X^{-b}
2. For $i \in [1, n]$ multiply by $\mathsf{TGSW}(X^{-a_i s_i})$

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

How to rotate by $-\varphi_{\mathbf{s}}(\mathbf{a}, b) = -b + \sum_{i=1}^{n} a_i s_i$?

1. Multiply by X^{-b}
2. For $i \in [1, n]$ multiply by $\mathsf{TGSW}(X^{-a_i s_i})$
   - $X^{a_i s_i} = 1 + (X^{a_i} - 1) \cdot s_i$, with $s_i \in \{0, 1\}$

Bootstrapping Algorithm (animation)

Bootstrapping algorithm of $(\mathbf{a}, b)$

1. Start from (a trivial) $\mathsf{TLWE}(v_0 + v_1 X + \cdots + v_{N-1} X^{N-1})$[a]
2. Rotate it by $p = -\varphi_{\mathbf{s}}(\mathbf{a}, b)$ positions
3. Extract the constant term (which encrypts v_p)

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

Rotate by p positions the coefficients $\mathbf{c} \in \mathsf{TLWE}$

- $(X^p \cdot \mathbf{c})$ when p is known
- $(\mathsf{TGSW}(X^p) \boxdot \mathbf{c})$ when p is unknown

How to rotate by $-\varphi_{\mathbf{s}}(\mathbf{a}, b) = -b + \sum_{i=1}^{n} a_i s_i$?

1. Multiply by X^{-b}
2. For $i \in [1, n]$ multiply by $\mathsf{TGSW}(X^{-a_i s_i})$
   - $X^{a_i s_i} = 1 + (X^{a_i} - 1) \cdot s_i$, with $s_i \in \{0, 1\}$
   - $\mathsf{TGSW}(X^{a_i s_i}) = h + (X^{a_i} - 1) \cdot \mathsf{TGSW}(s_i)$, where $\mathrm{BK} = \mathsf{TGSW}(s_i)$

Security analysis

Security analysis

Numerical security estimates

Based on [APS15],[LP11],[DM15] results

1. Convert the instance to a lattice problem
   - ✔ we tested: UniqueSVP, red to SIS, modSwitch...
2. Apply the best heuristics
3. Optimized all non-relevant parameters: $m, \varepsilon, q,$ trials ...

Security analysis

Numerical security estimates

Based on [APS15],[LP11],[DM15] results

1. Convert the instance to a lattice problem
   - ✔ we tested: UniqueSVP, red to SIS, modSwitch...
2. Apply the best heuristics
3. Optimized all non-relevant parameters: m, ε, q, trials . . .

Important security parameters

1. Noise rate: α
2. Entropy of the secret: n

and that's all!

- λ expressed solely as a function of (n, α)

Security parameter - the rainbow

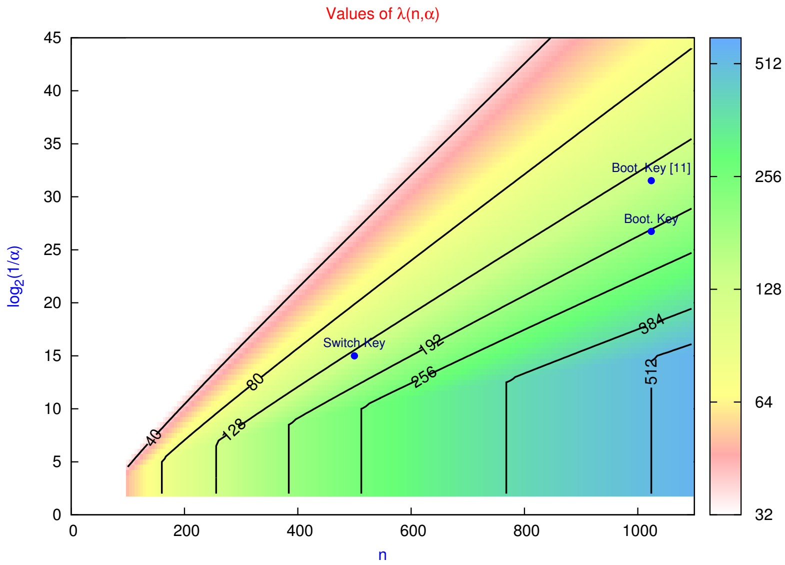


Table of contents

TFHE implementation

https://tfhe.github.io/tfhe/

TFHE implementation

https://tfhe.github.io/tfhe/

- Before: 1 bootstrapping in 52 ms

TFHE implementation

https://tfhe.github.io/tfhe/

- Before: 1 bootstrapping in 52 ms
- Now: 1 bootstrapping in 20 ms

Conclusion

Summary

- Construction and abstraction of TLWE and TGSW
- The external product $\boxdot : TGSW \times TLWE \rightarrow TLWE$
- Faster bootstrapping

Conclusion

Summary

- Construction and abstraction of TLWE and TGSW
- The external product $\boxdot : TGSW \times TLWE \rightarrow TLWE$
- Faster bootstrapping

More

- We can apply our results to leveled HE schemes
- We can improve this result and make FHE faster

Conclusion

Summary

- Construction and abstraction of TLWE and TGSW
- The external product $\boxdot : TGSW \times TLWE \rightarrow TLWE$
- Faster bootstrapping

More

- We can apply our results to leveled HE schemes
- We can improve this result and make FHE faster

Thank you!

Bibliography

- **[APS15]** Albrecht, M.R., Player, R., and Scott, S., *"On the concrete hardness of learning with errors."* Journal of Mathematical Cryptology 9.3 (2015): 169-203.
- **[BGV12]** Brakerski, Z., Gentry, C., and Vaikuntanathan, V. *"(Leveled) fully homomorphic encryption without bootstrapping."* In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (pp. 309-325). ACM (2012).
- **[BLPRS13]** Brakerski, Z., Langlois, A., Peikert, C., Regev, O., and Stehlé, D. *"Classical hardness of learning with errors."* In the proceedings of STOC'13 (2013).
- **[BP16]**, Brakerski, Z., and Perlman, R. *"Lattice-Based Fully Dynamic Multi-Key FHE with Short Ciphertexts."* In the proceedings of CRYPTO 2016 (2016).
- **[BR15]** Biasse, J-F., Ruiz, L., *"FHEW with Efficient Multibit Bootstrapping."* In the proceedings of LatinCrypt 2015 (2015).

- **[CS15]** Cheon, J.H., Stehlé, D., *"Fully Homomorphic Encryption over the Integers Revisited."* In the proceedings of EUROCRYPT'15. Springer-Verlag (2015).
- **[CGGI16]** Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. *"A Homomorphic LWE Based E-voting Scheme."* In International Workshop on Post-Quantum Cryptography (pp. 245-265). Springer International Publishing (2016).
- **[DM15]** Ducas, L., Micciancio, D., *"FHEW: Bootstrapping Homomorphic Encryption in less than a second."* In the proceedings of EUROCRYPT'15. Springer-Verlag (2015).
- **[GINX16]** Gama, N., Izabachene, M., Nguyen, P.Q., and Xie, X., *"Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions."* In the proceedings of EUROCRYPT'16. Springer-Verlag (2016).
- **[Gen09]** Gentry, C., *"A fully homomorphic encryption scheme [Ph. D. thesis]."* International Journal of Distributed Sensor Networks, Stanford University (2009).

Bibliography

- **[GSW13]** Gentry, C., Sahai, A., and Waters, B., *"Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based."* Advances in Cryptology–CRYPTO 2013. Springer Berlin Heidelberg, 2013. 75-92 (2013).
- **[LP11]** Lindner, R., and Peikert, C., *"Better key sizes (and attacks) for LWE-based encryption."* Cryptographers' Track at the RSA Conference. Springer Berlin Heidelberg (2011).
- **[LPR10]** Lyubashevsky, V., Peikert, C., and Regev, O., *"On Ideal Lattices and Learning with Errors over Rings."* Advances in Cryptology–EUROCRYPT 2010 (2010).
- **[Reg05]** Regev, O., *"On lattices, learning with errors, random linear codes, and cryptography."* In STOC, pp.84-93 (2005).