Circuit Privacy for FHEW/TFHE-Style Fully Homomorphic Encryption in Practice

Kamil Kluczniak

Fully Homomorphic Encryption - FHE

sk

$$c = \mathsf{Enc}(sk, m)$$

$$\xrightarrow{\quad c \quad}$$

$$\xleftarrow{\quad y \quad}$$

$$F(m) = \mathsf{Dec}(sk, y)$$

$$\mathsf{Eval}(c, F) = y$$

Applications

Formally

$$\mathsf{Setup}(\lambda) : sk, pk$$

$$\mathsf{Enc}(pk, m) : c$$

$$\mathsf{Dec}(sk, c) : m$$

$$\mathsf{Eval}(pk, f, \{c_1, \ldots, c_n\}) : c$$

Correctness

$$\Pr\left[\mathsf{Dec}(sk, c) \neq f(m_1, \ldots, m_n) : c = \mathsf{Eval}(pk, f, \{c_1, \ldots, c_n\}), m_i = \mathsf{Dec}(sk, c_i)\right] \leq \mathsf{negl}(\lambda)$$

Formally

$$\mathsf{Setup}(\lambda) : sk, pk$$

$$\mathsf{Enc}(pk, m) : c$$

$$\mathsf{Dec}(sk, c) : m$$

$$\mathsf{Eval}(pk, f, \{c_1, \ldots, c_n\}) : c$$

Compactness

$$\left|\mathsf{Eval}(pk, f, \{c_1, \ldots, c_n\})\right| = \mathsf{poly}(\lambda, |m|)$$

Formally

$$\mathsf{Setup}(\lambda) : sk, pk$$

$$\mathsf{Enc}(pk, m) : c$$

$$\mathsf{Dec}(sk, c) : m$$

$$\mathsf{Eval}(pk, f, \{c_1, \ldots, c_n\}) : c$$

IND-CPA

$b \leftarrow \{0,1\}$

$\mathsf{Enc}(sk, m_b) = c$

m_0, m_1 ←

c →

$\hat{b} \stackrel{?}{=} b$

Something is Missing

sk

$c = \mathsf{Enc}(sk, m)$

$F(m) = \mathsf{Dec}(sk, c_{out})$

c

c_{out}

$\mathsf{Eval}(pk, F, c) = c_{out}$

Something is Missing

sk

$c = \mathsf{Enc}(sk, m)$

c

$F(m) = \mathsf{Dec}(sk, c_{out})$

c_{out}

$\mathsf{Eval}(pk, F, c) = c_{out}$

Trivial Attack:

1. Guess the function F
2. Run Eval locally
3. Check if your output matches the servers output

DETERMINISTIC!!!
Leaks information on F!

Something is Missing

sk

$c = \mathsf{Enc}(sk, m)$

$F(m) = \mathsf{Dec}(sk, c_{out})$

c →

← c_{out}

$\mathsf{Eval}(pk, F, c) = c_{out}$

Trivial Attack:

1. Guess the function F
2. Run Eval locally
3. Check if your output matches the servers output

DETERMINISTIC!!!
Leaks information on F!

Circuit Privacy

$\mathsf{Eval}(pk, F, c)$

Has to be correct

c ←

→ c_{out}

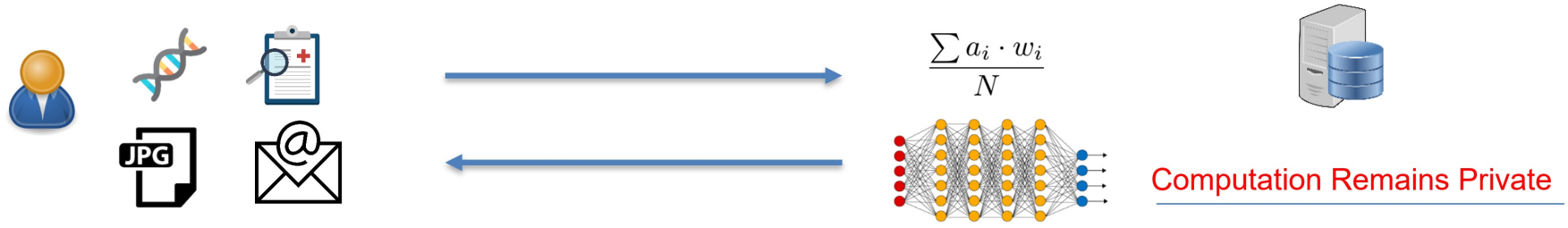
$\approx$

c →

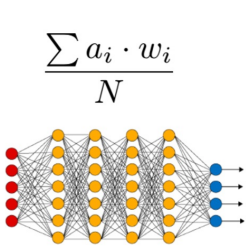
← c_{out}

$\mathsf{Sim}(pk, F(m))$

FHE With Circuit Privacy: Applications



FHE With Circuit Privacy: Applications



Computation Remains Private

Oblivious Pseudorandom Functions

sk

$c = \mathsf{Enc}(sk, m)$

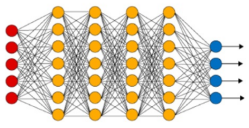
c

$c_{out} = \mathsf{Eval}(pk, \mathsf{PRF}(s, .), c)$

c_{out}

$\mathsf{PRF}(s, m) = \mathsf{Dec}(sk, c_{out})$

FHE With Circuit Privacy: Applications



Computation Remains Private

Blind Signatures

sk

$c = \mathsf{Enc}(sk, m)$

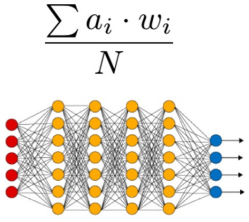
c

c_{out}

$\mathsf{SIGN}(s, m) = \mathsf{Dec}(sk, c_{out})$

$c_{out} = \mathsf{Eval}\big(pk, \mathsf{SIGN}(s, .), c\big)$

FHE With Circuit Privacy: Applications



Computation Remains Private

Secure Multi-Party Computation

sk

$c = \mathsf{Enc}(sk, m)$

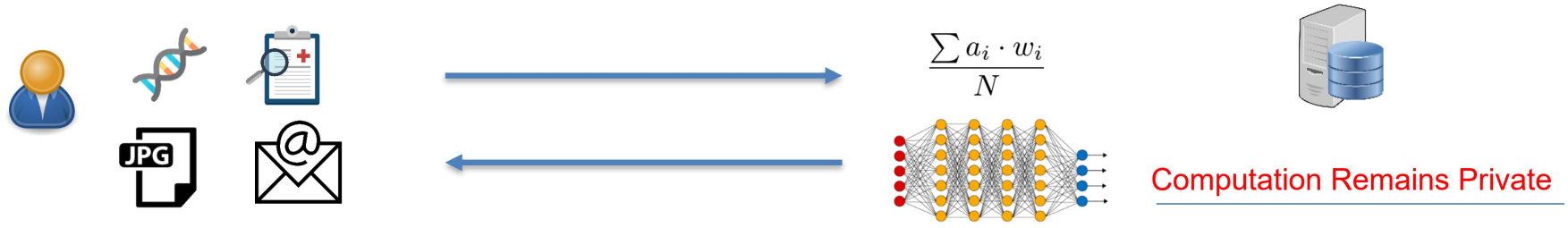
c

c_{out}

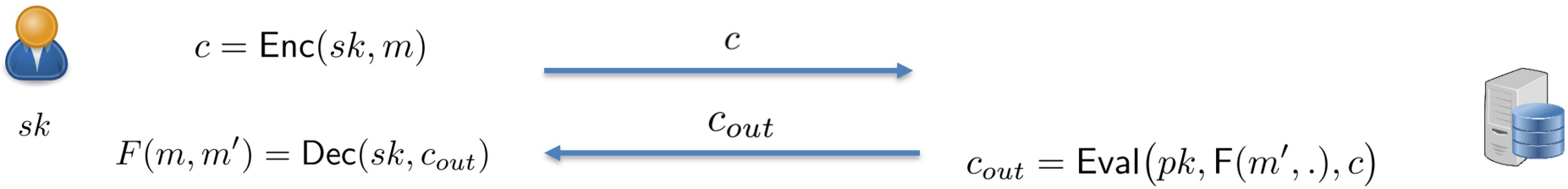
$F(m, m') = \mathsf{Dec}(sk, c_{out})$

$c_{out} = \mathsf{Eval}\big(pk, \mathsf{F}(m', .), c\big)$

FHE With Circuit Privacy: Applications



Secure Multi-Party Computation



Relaxed Notion of IND-CCA2 Security

[AGHV22]: Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. Achievable CCA2 relaxation for homomorphic encryption. TCC 2022

Frequently Asked Questions

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Q1: Does Computing a Universal Circuit Give Us Circuit Privacy?

Answer: No!

Q2: Does Adding a Freshly Sampled Ciphertext of Zero Give us Circuit Privacy?

Answer: Depends. But for LWE-based schemes the noise leaks information on the computation too.

Q3: Does Multikey FHE Give us Circuit Privacy?

Answer: Depends. You should be very careful...

See: Kamil Kluczniak, Giacomo Santato. On Circuit Private, Multikey and Threshold Approximate Homomorphic Encryption. IACR EPRINT 2023/301

Q4: If I homomorphically evaluate a neural network without circuit privacy does my entire model leak?

Answer: Doesn't seem to be so easy. But…. Who knows?

Contributions

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

1) New Ciphertext Sanitization Algorithm

- Provable Secure (No Heuristics!!!)
- Ciphertexts are Sanitized with only one (randomized) bootstrapping invocation without increasing the parameters too much
- Generalized and Tight Analysis of the Gaussian Leftover hash Lemma

2) First Practical Implementation of a FHEW/TFHE-Style Circuit-Private FHE Scheme

- Parameters and Implementation for floating point arithmetic (FFT) and finite field arithmetic (NTT)
- Different arithmetic = different modulus = different type of Gaussian sampling: all implemented and compared
- Optimizations that result in deterministic computation as fast as non-circuit private
- Python Scripts to Estimate and help choose circuit private parameters for the scheme

3) New Parameters and Estimation for State-of-The-Art Previous Work

- Parameters and Tests for Ducas-Stehle Washing Machine

4) New Library: FHE-Deck

- FHEW/TFHE-style functions/programmable bootstrapping with Circuit Privacy by Default

How to Get a Circuit Private FHE

Idea 1: Let's Add a Fresh Ciphertext of Zero

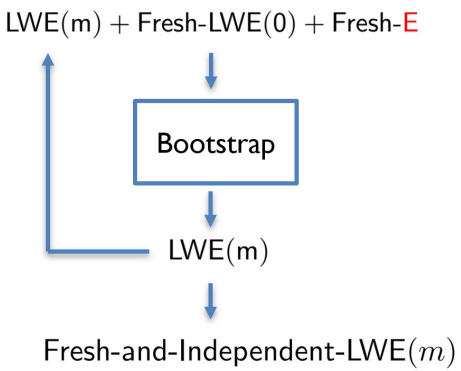
$$(\vec{a}, b = \langle \vec{a}, \vec{s} \rangle + m + e) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}} = \langle \vec{a}_{\text{fresh}}, \vec{s} \rangle + e_{\text{fresh}})$$

$$= (\vec{a}_{\text{out}} = \vec{a} + \vec{a}_{\text{fresh}}, b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + m + e + e_{\text{fresh}})$$

Idea 2: Let's Flood the Noise Term

$$(\vec{a}_{\text{out}}, b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + m + e + e_{\text{fresh}} + E)$$

Idea 3: Let's Add a Smaller Noise Term and Bootstrap Repeatedly

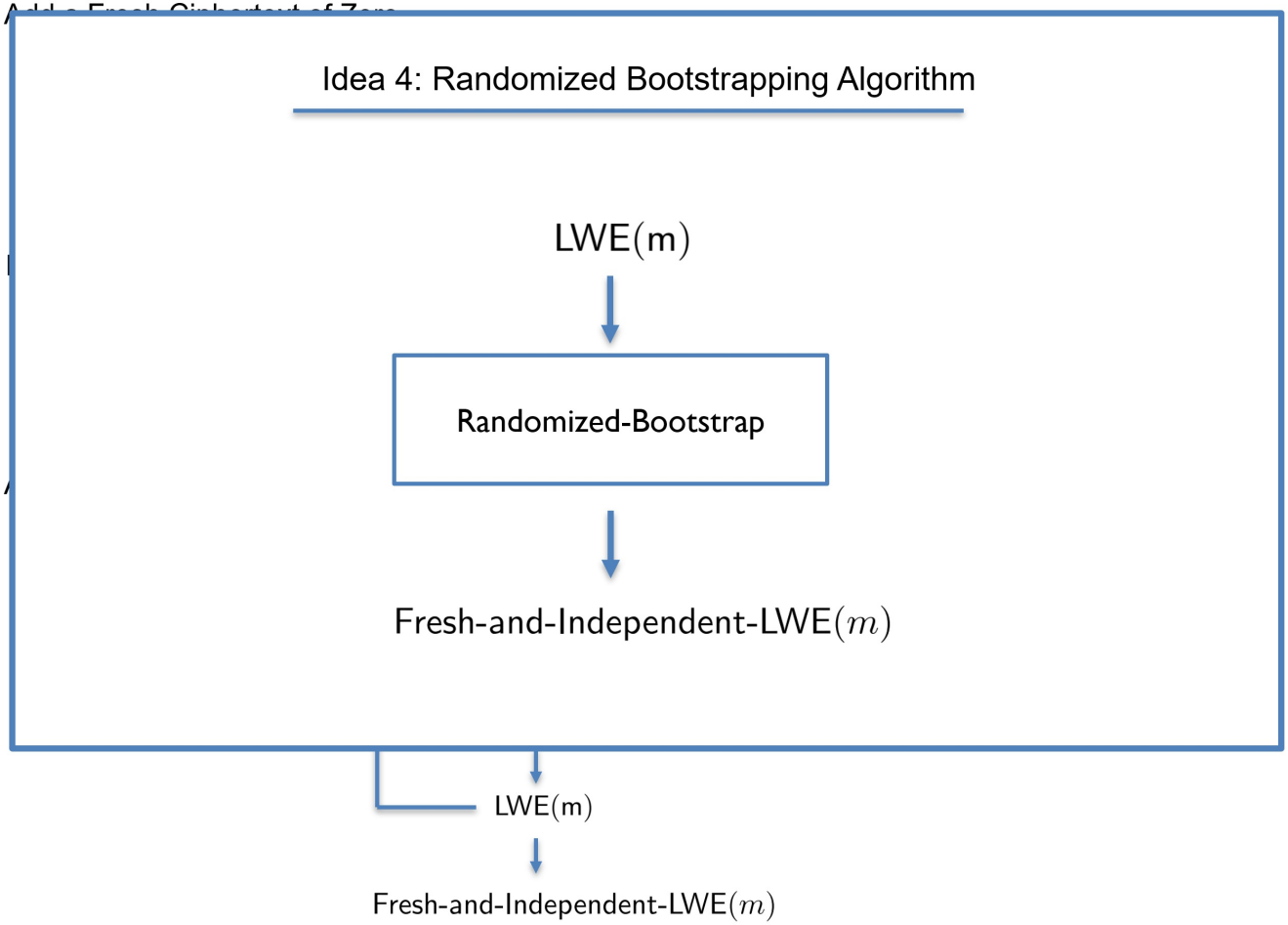


How to Get a Circuit Private FHE

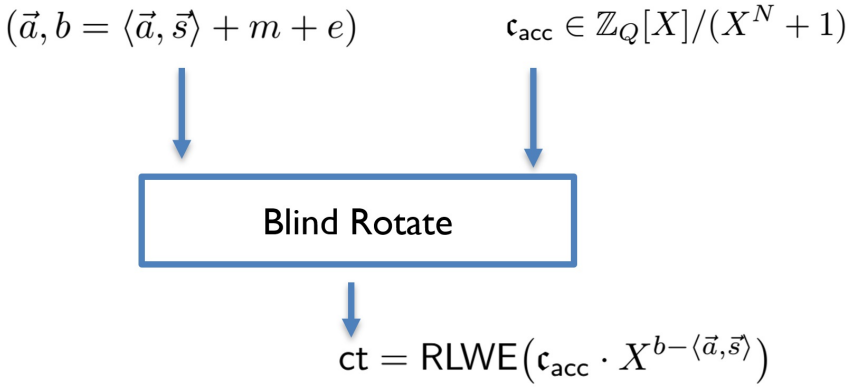
Idea 1: Let's Add a Fresh Ciphertext of Zero

Idea 2: Let's

Idea 3: Let's

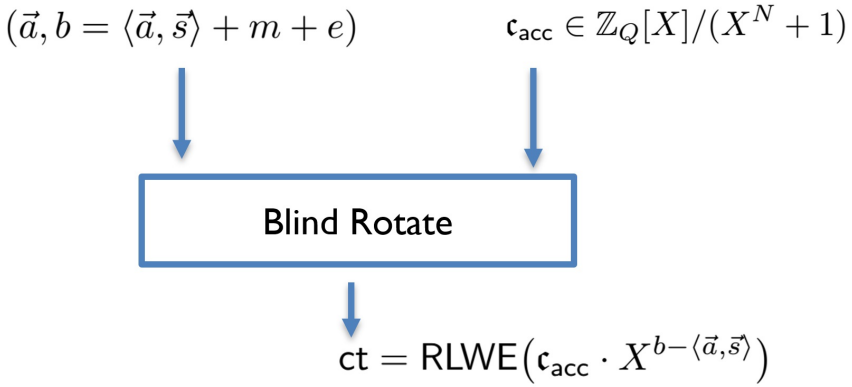


(Randomized) FHEW/TFHE-Style Bootstrapping



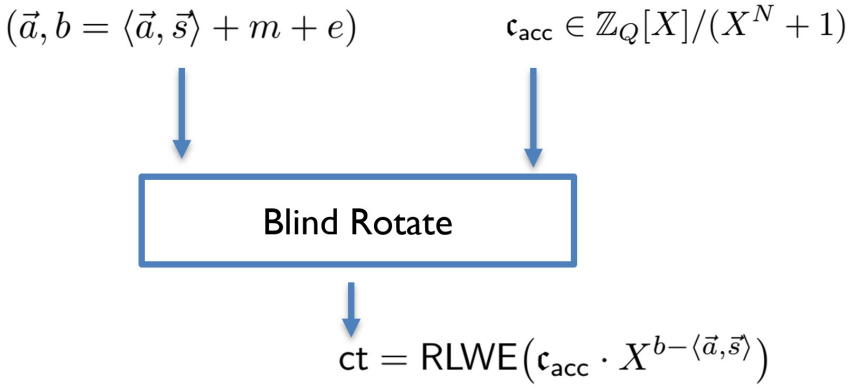
$\mathsf{Extract}(\mathsf{ct}) \to \mathsf{LWE}\big(\mathfrak{c}_{\mathsf{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}[1]\big)$

(Randomized) FHEW/TFHE-Style Bootstrapping



$\text{Extract}(\text{ct}) \to \text{LWE}\big(F(m)\big)$

(Randomized) FHEW/TFHE-Style Bootstrapping

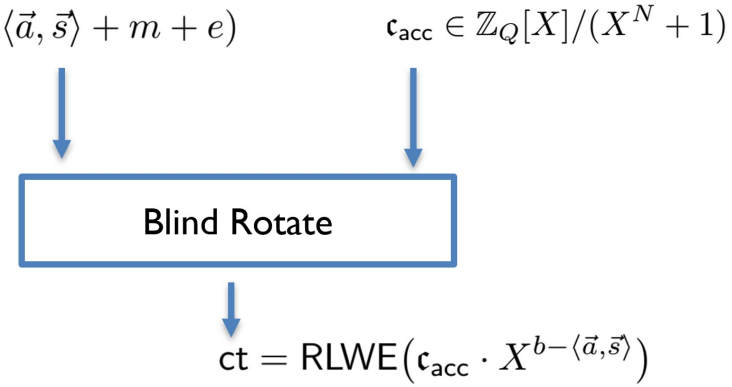


$\text{Extract}(\text{ct}) \to (\vec{a}_{\text{ext}}, b_{\text{ext}})$

where

$$b_{\text{ext}} = \langle \vec{a}_{\text{ext}}, \vec{s} \rangle + F(m) + e_{\text{ext}}$$

(Randomized) FHEW/TFHE-Style Bootstrapping



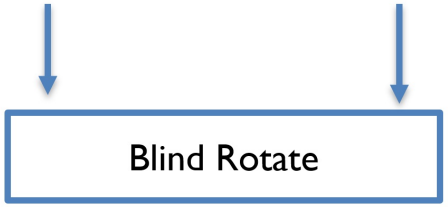
$$\text{Extract}(\text{ct}) \rightarrow (\vec{a}_{\text{ext}}, b_{\text{ext}}) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}}) = (\vec{a}_{\text{out}}, b_{\text{out}})$$

where

$$b_{\text{out}} = \underbrace{\langle \vec{a}_{\text{out}}, \vec{s} \rangle}_{\text{Uniform}} + F(m) + e_{\text{fresh}} + \underbrace{e_{\text{ext}}}_{\text{Depends on the input ciphertext}}$$

(Randomized) FHEW/TFHE-Style Bootstrapping

$(\vec{a}, b = \langle \vec{a}, \vec{s} \rangle + m + e)$ $\quad \mathfrak{c}_{\text{acc}} \in \mathbb{Z}_Q[X]/(X^N + 1)$



$\text{ct} = \text{RLWE}\left(\mathfrak{c}_{\text{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}\right)$

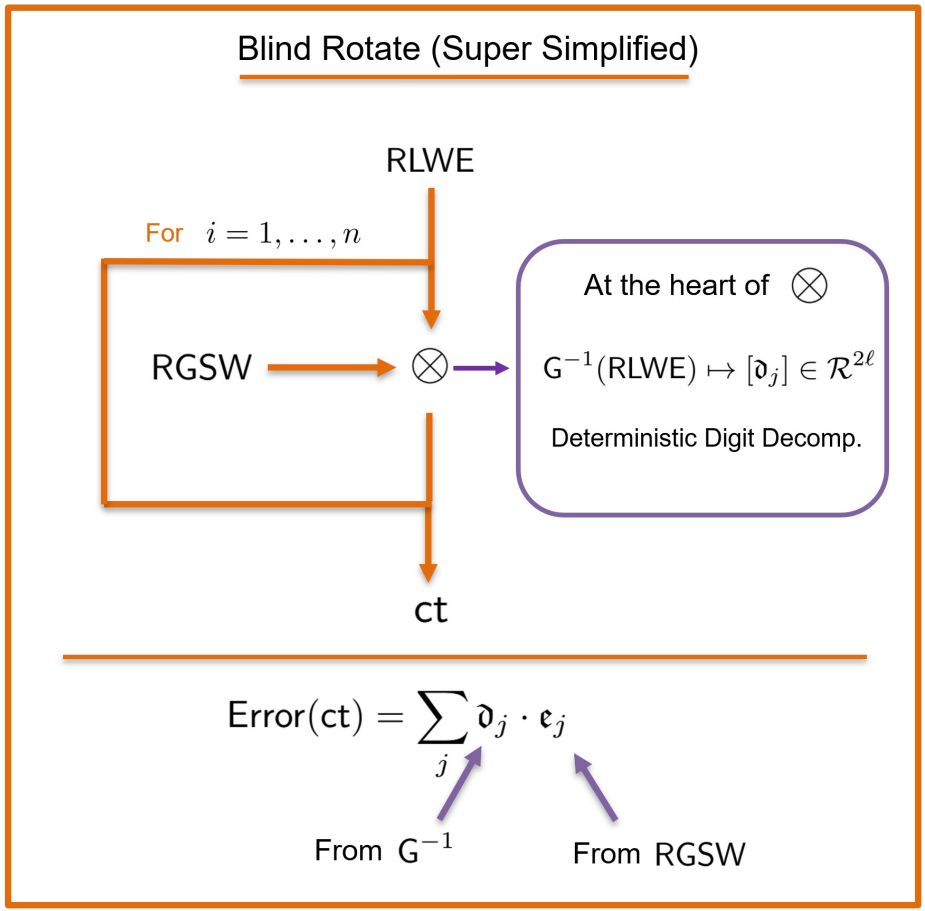
$\text{Extract}(\text{ct}) \to (\vec{a}_{\text{ext}}, b_{\text{ext}}) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}}) = (\vec{a}_{\text{out}}, b_{\text{out}})$

where

$$b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + F(m) + e_{\text{fresh}} + e_{\text{ext}}$$

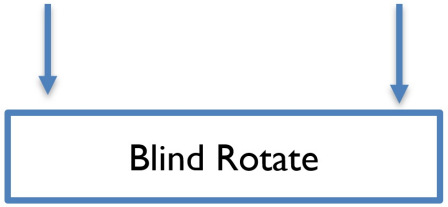
Uniform

Depends on the input ciphertext



(Randomized) FHEW/TFHE-Style Bootstrapping

$(\vec{a}, b = \langle \vec{a}, \vec{s} \rangle + m + e)$ $\quad \mathfrak{c}_{\text{acc}} \in \mathbb{Z}_Q[X]/(X^N + 1)$

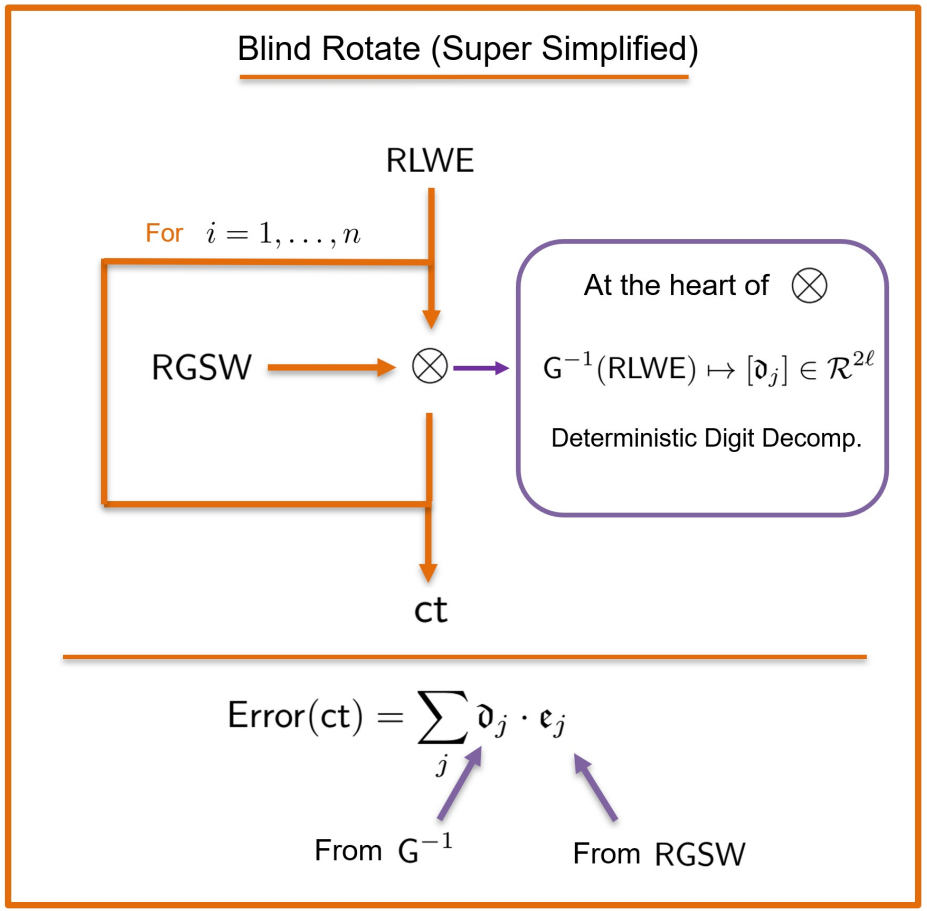


$\text{ct} = \text{RLWE}\left(\mathfrak{c}_{\text{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}\right)$

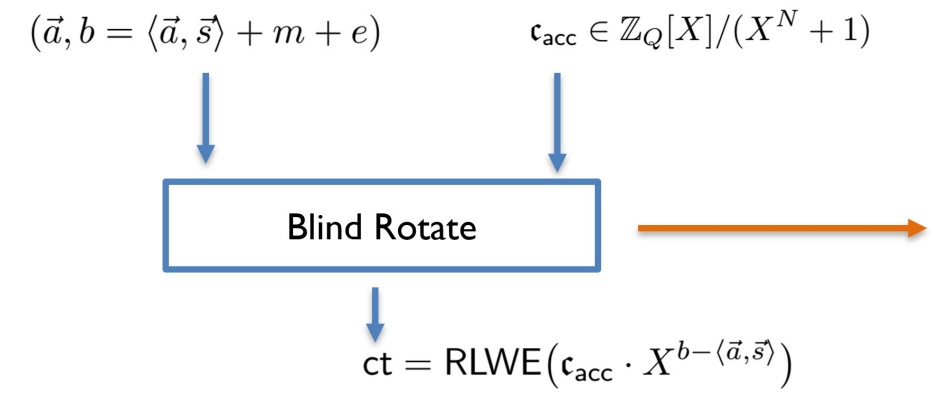
$\text{Extract}(\text{ct}) \rightarrow (\vec{a}_{\text{ext}}, b_{\text{ext}}) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}}) = (\vec{a}_{\text{out}}, b_{\text{out}})$

where

$$b_{\text{out}} = \underbrace{\langle \vec{a}_{\text{out}}, \vec{s} \rangle}_{\text{Uniform}} + F(m) + e_{\text{fresh}} + \underbrace{\sum_k d_k \cdot e_k}_{\text{Depends on the input ciphertext}}$$



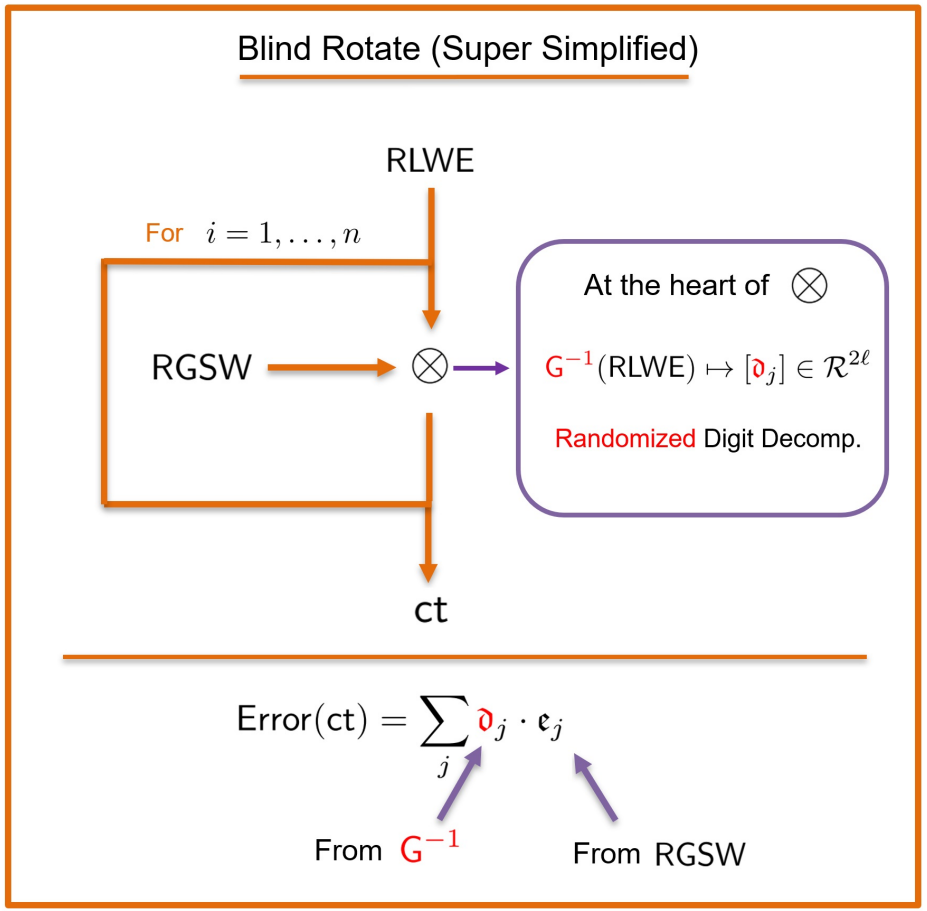
(Randomized) FHEW/TFHE-Style Bootstrapping



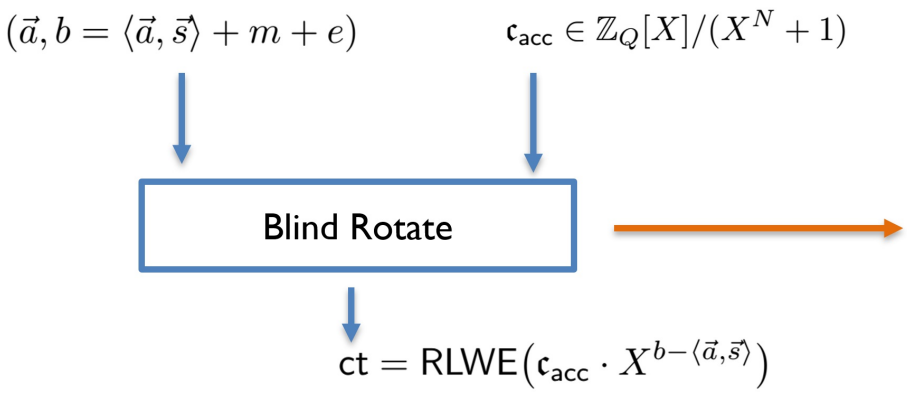
$$\text{Extract}(\text{ct}) \to (\vec{a}_{\text{ext}}, b_{\text{ext}}) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}}) = (\vec{a}_{\text{out}}, b_{\text{out}})$$

where

$$b_{\text{out}} = \underbrace{\langle \vec{a}_{\text{out}}, \vec{s} \rangle}_{\text{Uniform}} + F(m) + e_{\text{fresh}} + \underbrace{\sum_k d_k \cdot e_k}_{\text{Depends on the input ciphertext}}$$



(Randomized) FHEW/TFHE-Style Bootstrapping



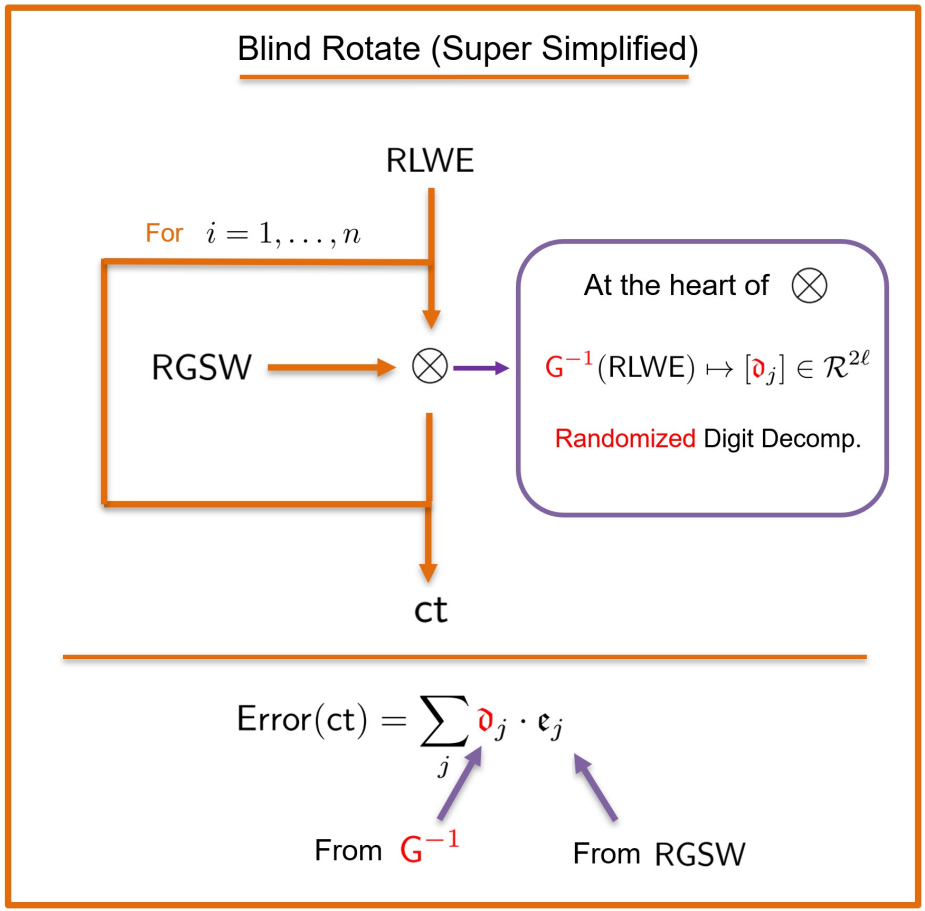
$$\text{Extract}(\text{ct}) \to (\vec{a}_{\text{ext}}, b_{\text{ext}}) + (\vec{a}_{\text{fresh}}, b_{\text{fresh}}) = (\vec{a}_{\text{out}}, b_{\text{out}})$$

where

$$b_{\text{out}} = \underbrace{\langle \vec{a}_{\text{out}}, \vec{s} \rangle}_{\text{Uniform}} + F(m) + e_{\text{fresh}} + \underbrace{\sum_k d_k \cdot e_k}_{\overset{s}{\approx} e_{\text{new}}}$$

1) Mask by fresh LWE sample

2) Independent Gaussian via G^{-1}



The Proof

$$b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + \mathfrak{c}_{\text{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}[1] + e_{\text{fresh}} + d + e_{\text{ext}}$$

$$b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + \mathfrak{c}_{\text{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}[1] + e_{\text{fresh}} + d + \sum_k d_k \cdot e_k$$

$$b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + F(m) + e_{\text{fresh}} + d + \sum_k d_k \cdot e_k$$

uniform random

$$b_{\text{out}} = \langle \vec{a}_{\text{out}}, \vec{s} \rangle + F(m) + e_{\text{fresh}} + e_{\text{new}}$$

1) Show that $e_{\text{ext}} = \sum_k d_k \cdot d_k$

2) Assume that $\mathfrak{c}_{\text{acc}} \cdot X^{b - \langle \vec{a}, \vec{s} \rangle}[1] = F(m)$ from **Correctness**

3) $\vec{a}_{\text{out}} = \vec{a}_{\text{ext}} + \vec{a}_{\text{fresh}}$ uniform random, because $\vec{a}_{\text{fresh}}$ is close to uniform from Leftover Hash Lemma

4) $d + \sum_k d_k \cdot d_k \overset{s}{\approx} e_{\text{new}}$ from Gaussian Leftover Hash Lemma (Generalized Lemma 3.6 from [BDPMW16])

[BDPMW16]: *Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. FHE Circuit Privacy Almost For Free. Crypto 2016*

Implementations, Tests and Comparisons

- Computational Security for IND-CPA: 128-bit

- Statistical Security for Circuit Privacy: 80-bit

- Ring: $\mathbb{Z}_Q[X]/(X^N+1)$ where $N = 2^{11}$

- Schemes: Our and [DS16]

- Implementations: Number Theoretic Transform (Intel Hexl) vs. Fast Fourier Transform (FFTW)

Parameters, Schemes and Implementations

Impl.	BK [MB]	Scheme	Bootstr.	$\log_2$(mod)	Mul-Det	Mul-Sim
NTT (Intel Hexl)	134	Our	1	48 (prime)	3	7
		[DS16]	5			
FFT (FFTW, 64-bit)	168	Our	1	36 (power-of-2)	4	10
		[DS16]	10			

[DS16]: Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. Eurocrypt 2016

The (Randomized) Decomposition Algorithm

Special modulus of the form $Q = b^k$

Used in FFT-Based Implementation

$\textsc{SampleG}(q = b^k, s, u)$
for $i = 0, \ldots, k-1$:
  $x_i \leftarrow \mathcal{D}_{b\mathbb{Z}+u,s}$
  $u := (u - x_i)/b \in \mathbb{Z}.$
return $(x_0, \ldots, x_{k-1})$.

NTT vs. FFT (Brief) Summary

NTT	Larger Mod	Complex G^{-1}
FFT	Numerical Errors	Simple G^{-1}

Arbitrary Modulus

Used in NTT-Based Implementation

$\textsc{SampleG}(s, \mathbf{u} = [u]_b^k, \mathbf{q} = [q]_b^k)$
  $\sigma := s/(b+1)$
  $\mathbf{p} \leftarrow \textsc{Perturb}(\sigma)$
  for $i = 0, \ldots, k-1$:
    $c_i := (c_{i-1} + u_i - p_i)/b$
  $\mathbf{z} \leftarrow \textsc{SampleD}(\sigma, \mathbf{c})$
  for $i = 0, \ldots, k-2$:
    $t_i := b \cdot z_i - z_{i-1} + q_i \cdot z_{k-1} + u_i$
  $t_{k-1} := q_{k-1} \cdot z_{k-1} - z_{k-2} + u_{k-1}$
  return t

$\textsc{Perturb}(\sigma)$
  $\beta := 0$
  for $i = 0, \ldots, k-1$:
    $c_i := \beta/l_i$, and $\sigma_i := \sigma/l_i$
    $z_i \leftarrow \lfloor c_i \rfloor + \textsc{SampleZ}_t(\sigma_i, \lfloor c_i \rceil_{[0,1)}, s)$
    $\beta := -z_i h_i$
  $p_0 := (2b+1)z_0 + bz_1$
  for $i := 1, \ldots, k-1$:
    $p_i := b(z_{i-1} + 2z_i + z_{i+1})$
  return p

$\textsc{SampleD}(\sigma, \mathbf{c})$
  $z_{k-1} \leftarrow \lfloor -c_{k-1}/d_{k-1} \rfloor$
  $z_{k-1} \leftarrow z_{k-1} + \textsc{SampleZ}_t(\sigma/d_{k-1}, \lfloor -c_{k-1}/d_{k-1} \rceil_{[0,1)}, s)$
  $\mathbf{c} := \mathbf{c} + z_{k-1}\mathbf{d}$
  for $i \in \{0, \ldots, k-2\}$:
    $z_i \leftarrow \lfloor -c_i \rfloor + \textsc{SampleZ}_t(\sigma, \lfloor -c_i \rceil_{[0,1)}, s)$
  return z

[DS16]: Nicholas Genise and Daniele Micciancio. Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus. Eurocrypt 2018

Implementations, Tests and Comparisons

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Parameters, Schemes and Implementations

Impl.	BK [MB]	Scheme	Bootstr.	$\log_2$(mod)	Mul-Det	Mul-Sim
NTT (Intel Hexl)	134	Our	1	48 (prime)	3	7
		[DS16]	5			
FFT (FFTW, 64-bit)	168	Our	1	36 (power-of-2)	4	10
		[DS16]	10			

Tests and Comparison

Impl.	Scheme	Total [s] det	Total [s] simul	Bootstrap [s] det	Bootstrap [s] simul	G^{-1}
NTT	Ours	0.14	1.44	0.14	1.44	78%
	[DS16]		3.55		0.71	–
FFT	Ours	0.27	1.22	0.27	1.22	42%
	[DS16]		6.38		0.63	–

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

FHE-Deck: Circuit Privacy by Default

https://github.com/fhe-deck

- Implementations of (different types of functional/PBS) variants of **FHEW/TFHE/NTRUnium/FINAL** schemes
- All parameters sets support **Circuit Privacy**
- User doesn't have to think much about it
- …. and estimator scripts to choose new parameter sets

Example

```
lwe_ct ct2 = context.encrypt(2);
lwe_ct mct3 = context.encrypt(-3);

auto fun_relu = [](long m) -> long {
    if(m >= 0){
        return m;
    }else{
        return 0;
    }
};

lwe_ct ct_relu = context.eval_lut(&ct2, fun_relu);
std::cout << "Decrypt(ct_relu): " << context.decrypt(&ct_relu) << std::endl;
ct_relu = context.eval_lut(&mct3, fun_relu);
std::cout << "Decrypt(ct_relu): " << context.decrypt(&ct_relu) << std::endl;

std::ofstream fout("filename.txt");
fout << ct_relu;
```

Decrypt(ct_relu): 2

Decrypt(ct_relu): 0

Sanitizes the Ciphertext

Thank You

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

FHE-Deck: Circuit Privacy by Default

https://github.com/fhe-deck

(Newest Version will be Available Next Week)

Circuit Privacy for FHEW/TFHE-Style Fully Homomorphic Encryption in Practice

Kamil Kluczniak

https://eprint.iacr.org/2022/1459.pdf

On Circuit Private, Multikey and Threshold Approximate Homomorphic Encryption

Kamil Kluczniak and Giacomo Santato

https://eprint.iacr.org/2023/301.pdf