Secure numerical computations using fully homomorphic encryption

Michael Schlottke-Lakemper, Arseniy Kholod

High-Performance Scientific Computing, University of Augsburg, Germany

JuliaCon 2024, Eindhoven, Netherlands, 10^{th} July 2024

Acknowledgments

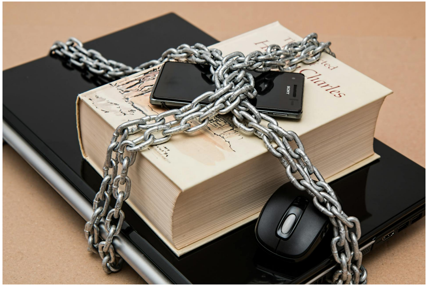
Arseniy Kholod

Repro: `github.com/hpsc-lab/talk-2024-juliacon-secure_numerical_computations`

Data security/privacy is all the rage

- Security: no *unauthorized* access
- Privacy: no *unnecessary* access
- Examples:
  - Health data
  - Financial information
  - Proprietary algorithms
  - Intellectual property

$\Rightarrow$ Major challenge for cloud computing



How to balance security/privacy and functionality?

Desire to keep data private | Need to keep algorithms private

Zero trust computing

Idea: only process *encrypted* data in the cloud

$$2 \quad + \quad 4 \quad = \quad 6$$

Idea: only process *encrypted* data in the cloud

$$
\begin{array}{ccccc}
2 & + & 4 & = & 6 \\
\underbrace{2+5}_{7} & + & \underbrace{4+8}_{12} & = & \underbrace{6+13}_{19}
\end{array}
$$

random secret | encrypted data

Idea: only process *encrypted* data in the cloud

$$2 \qquad + \quad 4 \qquad = \quad 6$$

$$\underbrace{2+5}_{7} \quad + \quad \underbrace{4+8}_{12} \quad = \quad \underbrace{6+13}_{19}$$

random secret | encrypted data

- Problem 1: Requires local knowledge about exact algorithm
- Problem 2: Does not save computations locally

Idea: only process *encrypted* data in the cloud

$$\begin{array}{ccccc} 2 & + & 4 & = & 6 \\ \underbrace{2+5}_{7} & + & \underbrace{4+8}_{12} & = & \underbrace{6+13}_{19} \end{array}$$

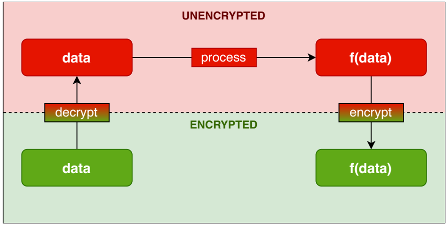
random secret | encrypted data

- Problem 1: Requires local knowledge about exact algorithm
- Problem 2: Does not save computations locally
- Problem 3: It gets worse...

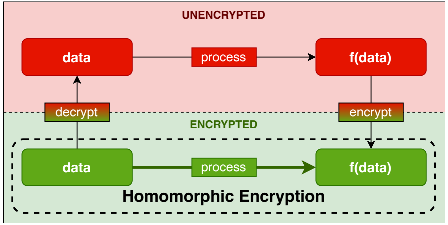
$$\underbrace{2+5}_{7} \cdot \underbrace{4+8}_{12} = \underbrace{8 + 2 \cdot 8 + 5 \cdot 4 + 5 \cdot 8}_{84}$$

What is homomorphic encryption (HE)?

- Perform computations on encrypted data
- Result identical to plaintext computation
- Fully homomorphic encryption (FHE): unbounded algorithmic depth
- Data/algorithm mutually unknown
  $\rightarrow$ zero trust computing



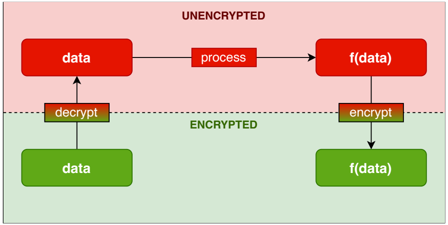
Without HE



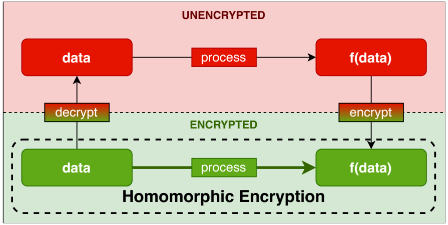
With HE

What is homomorphic encryption (HE)?

- Perform computations on encrypted data
- Result identical to plaintext computation
- Fully homomorphic encryption (FHE): unbounded algorithmic depth
- Data/algorithm mutually unknown
  $\rightarrow$ zero trust computing
- Drawbacks
  - Limited basic operations
  - Increased resource requirements
  - Possibly reduced accuracy



Without HE



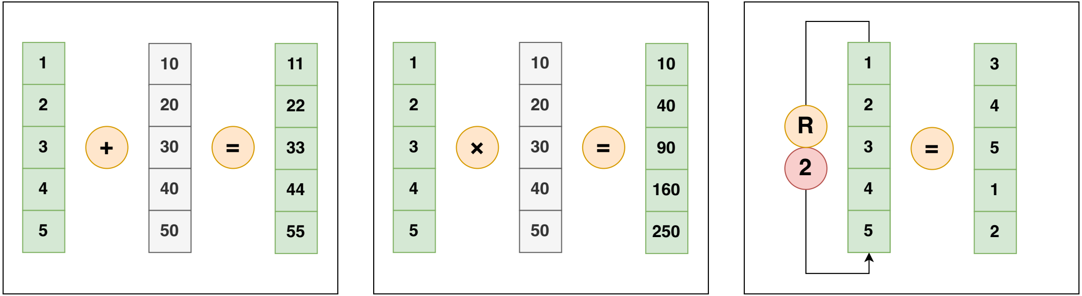
With HE

Fundamental concepts of FHE (by a non-cryptographer)

- Asymmetric (public key) encryption
- Schemes for integers: BFV, BGV, TFHE, ...
- Schemes for real numbers: CKKS
- Lattice-based/Ring Learning With Errors (RLWE)
- Principle: "hide data behind noise"
- Challenge: noise grows with algorithmic depth
  $\rightarrow$ requires bootstrapping

FHE.org community

Building blocks for homomorphically encrypted algorithms



Add vectors Multiply vectors Rotate vectors

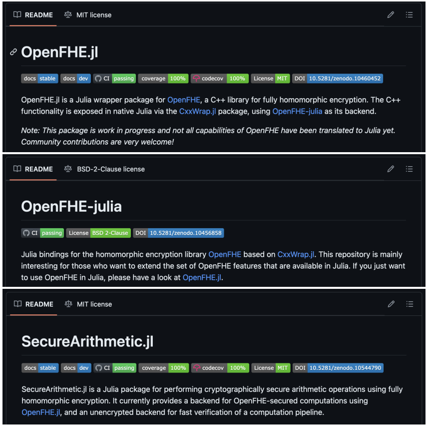
- Element-wise vector operations for efficiency
- Plus: plaintext operations with vectors/scalars
- Plus: bootstrapping to reduce noise

Homomorphic encryption in Julia

- RAMPARTS (2017–2019, Archer et al.)
- Paillier.jl (2020, Brian Thorne)
- ToyFHE.jl (2020, Keno Fischer)
- SEAL.jl (2021, MSL)
- OpenFHE.jl (2023–, AK & MSL)
- SecureArithmetic.jl (2023–, AK & MSL)

OpenFHE, OpenFHE.jl, and SecureArithmetic.jl

- OpenFHE: open-source FHE library
- OpenFHE.jl: Julia wrapper for OpenFHE
- OpenFHE-julia: Julia/C++ bindings using CxxWrap.jl
- SecureArithmetic.jl: convenience functions / syntatic sugar for rapid prototyping



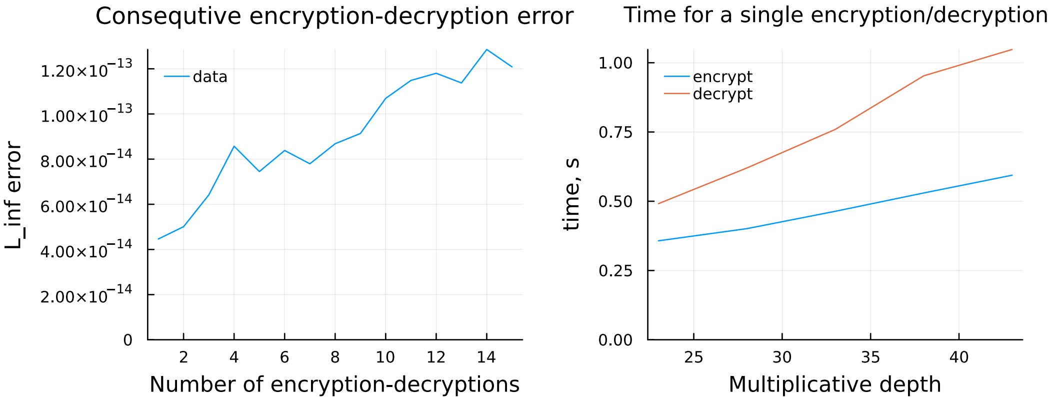
`github.com/openfheorg/openfhe-development`

In practice: need to set up a secure context

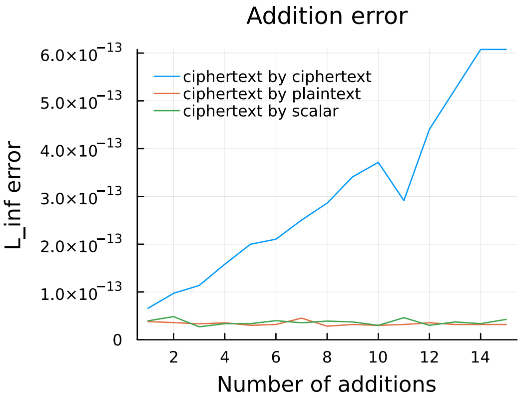
- Need to choose parameters related to
  - Problem size
  - Encryption scheme
  - Multiplicative depth
  - FHE library
- Parameters strongly affect accuracy and performance
- System for performance analysis:
  - AMD Ryzen Threadripper 3990X
  - 256 GiB memory

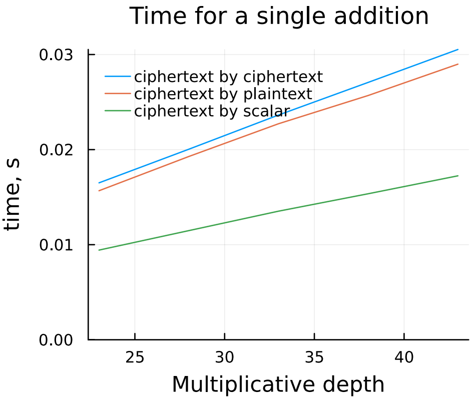
Parameter	**Value**
Security level	128 Bits
Scaling modulus	59 Bits
First modulus	60 Bits
Ring dimension	2^{17}
Batch size	*various*
Multiplicative depth	25 (mostly)

Accuracy and performance of fundamental operations: encrypt/decrypt

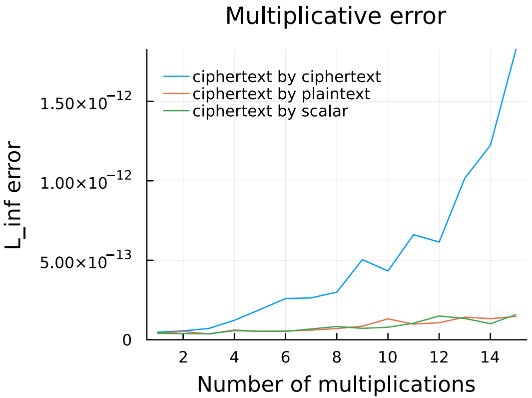


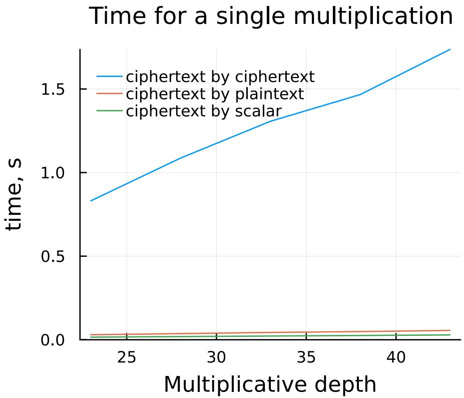
Accuracy and performance of fundamental operations: add



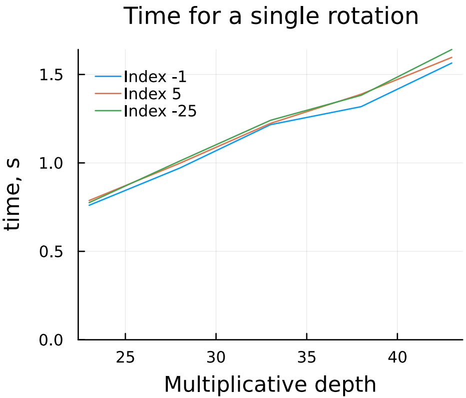


Accuracy and performance of fundamental operations: multiply

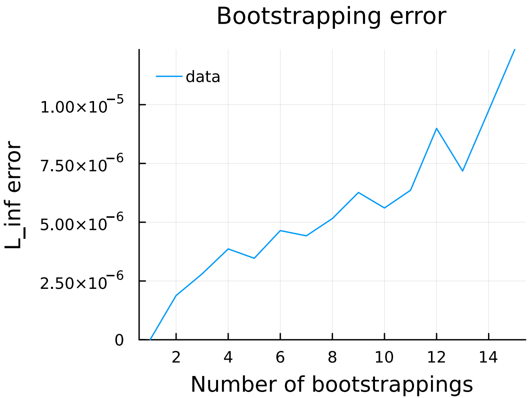


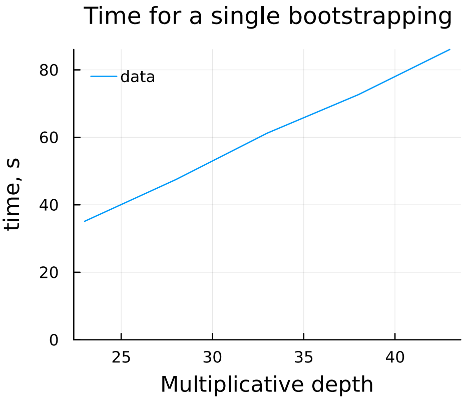


Accuracy and performance of fundamental operations: rotate



Accuracy and performance of fundamental operations: bootstrap





Accuracy and performance of fundamental operations: overview

OpenFHE operation	**Error**	**Time**
Encrypt(plaintext)	$\approx 10^{-1}$	$\approx 10^{1}$
Decrypt(ciphertext)	$\approx 10^{-1}$	$\approx 10^{1}$
EvalAdd(ciphertext, scalar)	$\approx 10^{-1}$	$\approx 10^{0}$
EvalAdd(ciphertext, plaintext)	$\approx 10^{-1}$	$\approx 10^{0}$
EvalAdd(ciphertext, ciphertext)	1	1
EvalMult(ciphertext, scalar)	$\approx 10^{0}$	$\approx 10^{0}$
EvalMult(ciphertext, plaintext)	$\approx 10^{0}$	$\approx 10^{0}$
EvalMult(ciphertext, ciphertext)	$\approx 10^{1}$	$\approx 10^{1}$
EvalRotate(ciphertext)	$\approx 10^{-1}$	$\approx 10^{1}$
EvalBootstrap(ciphertext)	$\approx 10^{7}$	$\approx 10^{3}$

Example: secure numerical simulation

Linear advection equation 1D:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \quad a > 0 \tag{1}$$

Linear advection equation 2D:

$$\frac{\partial u}{\partial t} + a_x\frac{\partial u}{\partial x} + a_y\frac{\partial u}{\partial y} = 0, \quad a_x, a_y > 0 \tag{2}$$

Numerical method based on finite-difference approximations

First-order upwind scheme (1D):

$$u_i^{n+1} = u_i^n - \frac{a\Delta t}{\Delta x}(u_i^n - u_{i-1}^n) \tag{3}$$

Second-order Lax-Wendroff scheme (1D):

$$u_i^{n+1} = u_i^n - a\frac{\Delta t}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) + a^2\frac{\Delta t^2}{2\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{4}$$

Rewrite numerical method in FHE operations: Lax-Wendroff scheme (1D)

Original formulation:

$$u_i^{n+1} = u_i^n - a\frac{\Delta t}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) + a^2\frac{\Delta t^2}{2\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{5}$$

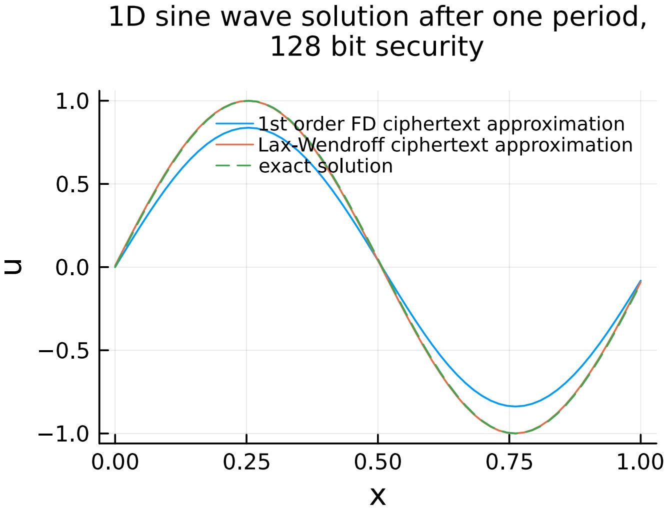
FHE formulation:

$$\begin{aligned} \boldsymbol{u^{n+1}} &= \boldsymbol{u^n} - a\frac{\Delta t}{2\Delta x}(\text{circshift}(\boldsymbol{u^n}, -1) - \text{circshift}(\boldsymbol{u^n}, 1)) \\ &\quad + a^2\frac{\Delta t^2}{2\Delta x^2}(\text{circshift}(\boldsymbol{u^n}, -1) - 2\boldsymbol{u^n} + \text{circshift}(\boldsymbol{u^n}, 1)) \end{aligned} \tag{6}$$

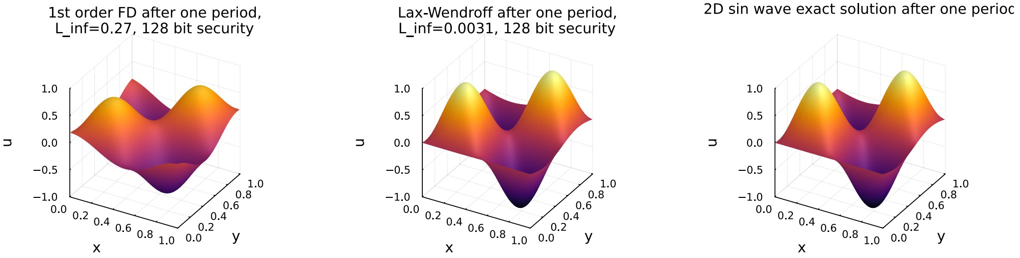
Julia code:

```
u = u -   r/2 * (circshift(u, -1)       - circshift(u, 1))
      + r^2/2 * (circshift(u, -1) - 2*u - circshift(u, 1))
```

Results for secure numerical simulation: 1D



Results for secure numerical simulation: 2D



Convergence analysis for first- and second-order schemes

1D

1st order FD

N_x	$L2_{mean}$	EOC
8	0.1532	-
16	0.076	1.011
32	0.0379	1.004
64	0.0189	1.004
mean	-	1.006

Lax-Wendroff

N_x	$L2_{mean}$	EOC
8	0.061	-
16	0.0147	2.053
32	0.0036	2.03
64	0.0009	2.0
mean	-	2.028

2D

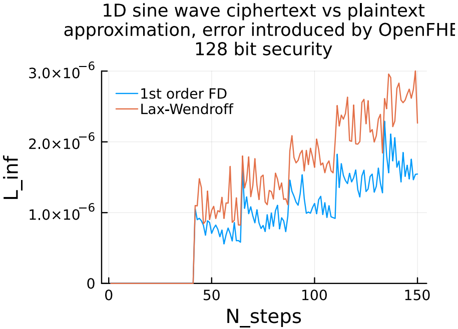
1st order FD

$N_x = N_y$	$L2_{mean}$	EOC
8	0.201	-
16	0.1123	0.84
32	0.0598	0.909
64	0.031	0.948
mean	-	0.899

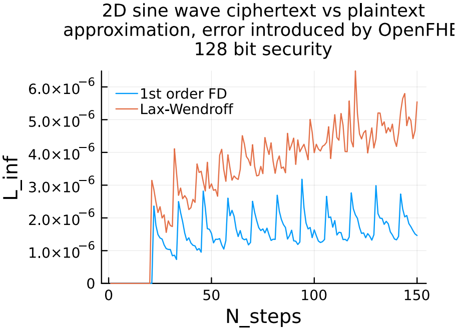
Lax-Wendroff

$N_x = N_y$	$L2_{mean}$	EOC
8	0.0365	-
16	0.0071	2.362
32	0.0016	2.15
64	0.0004	2.0
mean	-	2.171

Error analysis compared to plaintext computation

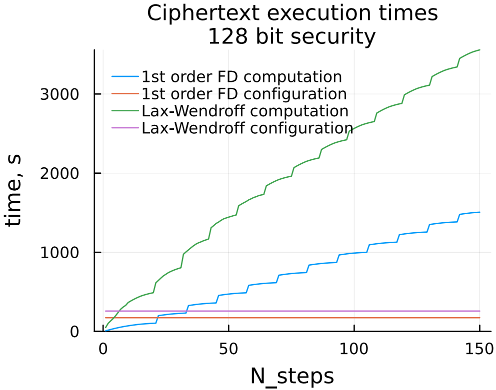


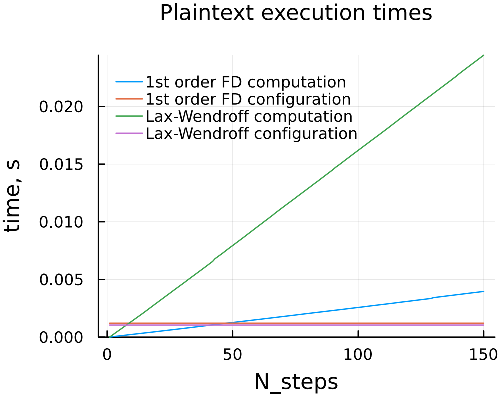
1D



2D

Performance analysis for secure numerical simulation in 2D





Summary and outlook

- FHE: (arbitrary) algorithms over encrypted data
- Practical limitations (complexity, runtime, FHE library)
- Simple experimentation with SecureArithmetic.jl
- Secure numerical simulations is feasible
- Next: more complexity, new applications/schemes

Repro repo

github.com/hpsc-lab/talk-2024-juliacon-secure_numerical_computations

Summary and outlook

- FHE: (arbitrary) algorithms over encrypted data
- Practical limitations (complexity, runtime, FHE library)
- Simple experimentation with SecureArithmetic.jl
- Secure numerical simulations is feasible
- Next: more complexity, new applications/schemes

Repro repo

`github.com/hpsc-lab/talk-2024-juliacon-secure_numerical_computations`

Thank you for your attention!

Backup

The CKKS scheme

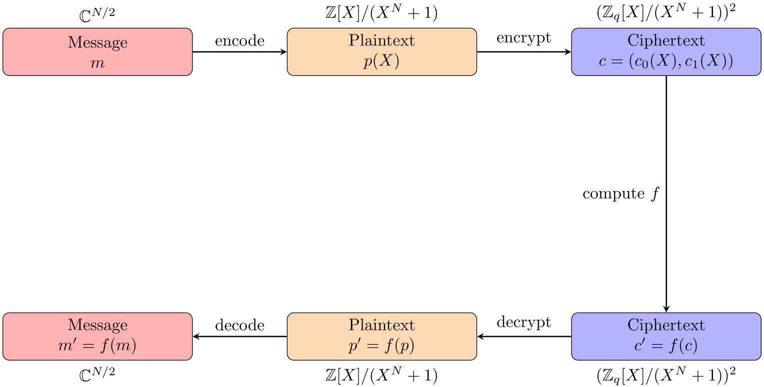


Image: Daniel Huynh

`https://blog.openmined.org/ckks-explained-part-1-simple-encoding-and-decoding/`

Memory boundedness of FHE operations