PYFHEL
PYthon For Homomorphic Encryption Libraries

Alberto Ibarrondo

Alexander Viand

IDEMIA

EURECOM
Sophia Antipolis

ETH zürich

Agenda

→ 1 **Why a Python wrapper?**
- Nicer API
- Nicer Language

→ 2 **Why another Python wrapper?**
- Improvements
- Teaching

→ 3 **Architecture & Design**

→ 4 **Demo Time!**

→ 5 **Conclusion**

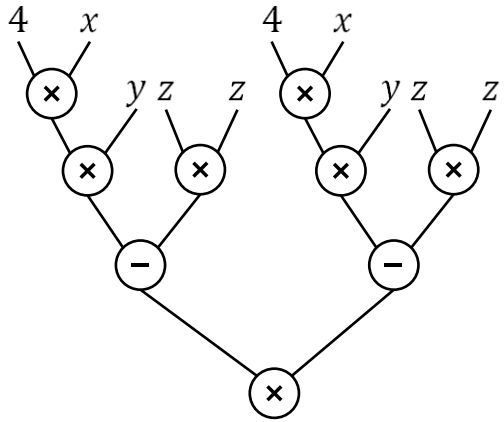
Why a Python wrapper?

1. Nicer API
   Nicer Language

1.1 Nicer API

- Most of the canon FHE libraries (**SEAL**, PALISADE, HElib) are written with a functional approach, missing convenient operator overloads (*, +, -):

$$\alpha = (4xy - z^2)^2$$




```
var t0 = 4*x;
var t1 = t1*y;
var t2 = z*z;
var t3 = t1-t2;
return t3*t3;
```

Ideal code

```
fhe.mul_plain_inp(x,4)
fhe.mul_inp(x,y);
fhe.square_inp(z,z);
fhe.sub_inp(x,z);
fhe.square_inp(x,x);
return x;
```

Realistic code

- Existing API (plain, in-place ops) is driven by how operations differ in implementation, not by how they're used.

1.2 Nicer Language: Python

- Most of the canon FHE libraries (**SEAL**, PALISADE, HElib) are written in **C++**
  - Not particularly friendly for newcomers
  - No unified compilation toolchain
  - But…**FAST**!

- Enter **Python**
  - The second most popular full programming language [1] (just below Javascript)
  - → Much more widespread: targets a wider audience
  - → Newcomer friendly. Sometimes it even looks like **pseudo-code!**.
  - More accessible: unified compilation/installation toolchain (pip install myrepo)
  - Especially relevant for data domains: data science & engineering, Machine Learning
  - But…**SLOW**!

FHE is already orders of magnitude slower

Python with C++ speed?

(1) Stackoverflow 2021 survey: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language

Why another Python wrapper?

2

Improvements

Teaching

2.1 Improvements

- **Python at C++ speed:**
  - FHE libraries based on native Python types are slower. (**pyFHE**).
  - Automatic C++ wrapping tools like *pybind11* or *Boost.Python* require large parts of the wrapper to be written in C++ to preserve performance (**PySEAL**, **TenSEAL**).

- **Seamless compilation:**
  - Standard Wrappers:

  ⟶ Precompiled binaries for each version/system (**TenSEAL**)

  ⟶ Compilation toolchain only in one OS (**SEAL-Python**)
  - Our system: Actually compile from python (can be generalized to other projects!)

- Expose **underlying features** that don't have a pretty API in **SEAL**:
  - Working directly on Polynomials.
  - Memory management, keeping track of sizes/pointers/etc.

2.2 Suitable for FHE Teaching

- FHE is establishing its presence in the CS curriculum
  - "An Intensive Introduction to Cryptography" (Harvard CS 127, Boaz Barak)
  - "Applied Cryptography" (ETH Zurich 263-4660, Kenny Paterson)
  - "Advanced Cryptography" (Princeton COS 533, Mark Zhandry)
  - …
- Practical exercises require a simple interface and an exploration-friendly playground.
  - Python is dynamic! You can play with existing objects and functions at **runtime**
  - Lots of courses use Python already (including for auto-grading systems)
- **Low-level access to Polynomials enables more interesting exercises**
  - `seal::Evaluator` interface allows little beyond implementing FHE applications
  - SEAL uses a very low-level representation to work on polynomials (No abstraction below `Ctxt`)
  - Student implementations of basic schemes:
  - Understanding crypto requires "breaking things" (e.g., implementing Li-Micciancio attack)
  - Allows (re-)implementing core algorithms (Poly ↔ Numpy conversion allows easy verification)

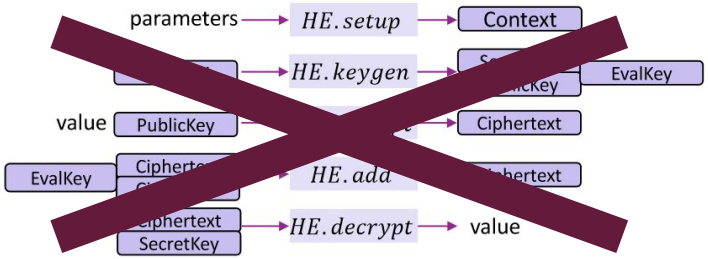
Architecture & Design

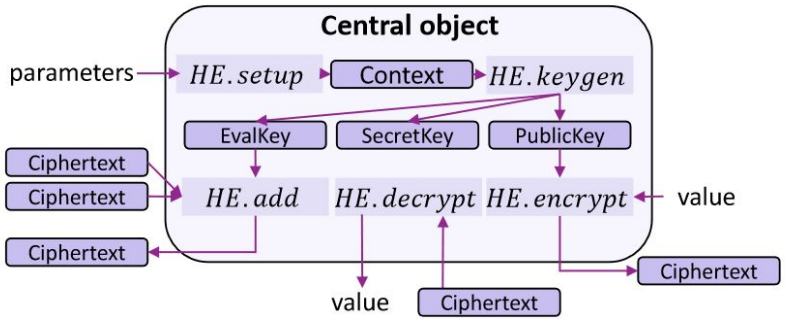
3

3.1. Design Principles

- **One-click install:** pip install Pyfhel
  - Not precompiled versions (*TenSeal*), but actually the source code
  → Can benefit from local compiler optimizations!
  - Installs CMake under the hood from a pip repo, and uses it for cmake-based libraries (SEAL ☺).
  - Uses the underlying Python compiler (GCC in Linux, MSVC for windows) to compile everything.

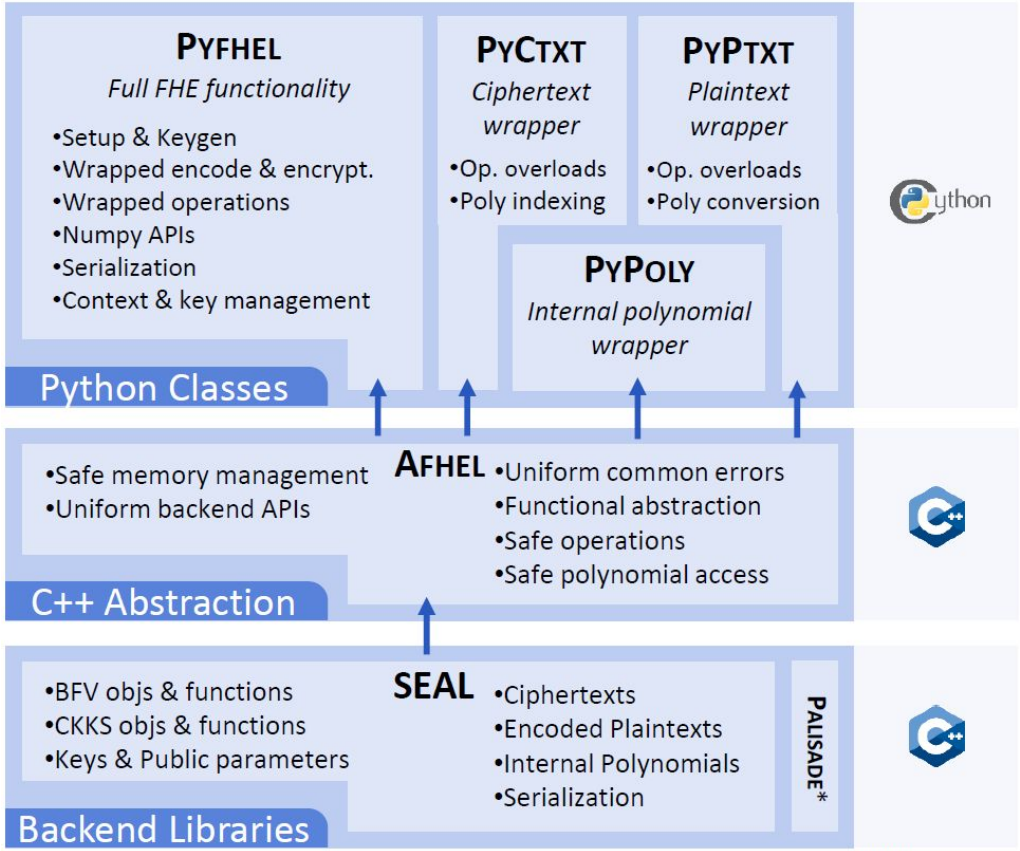
- **~~Functional~~ Centralized approach**





- **C++ to abstract classes & Cython to move it to Python**

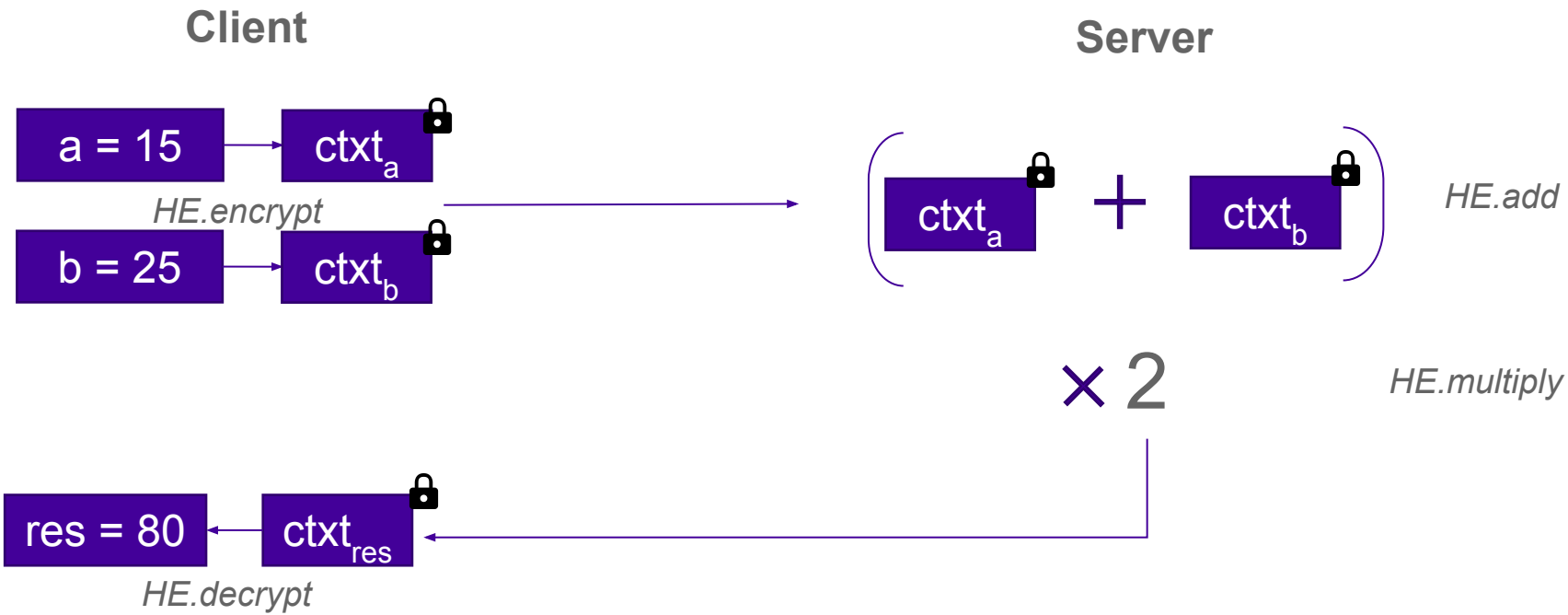
3.2. Architecture of Pyfhel



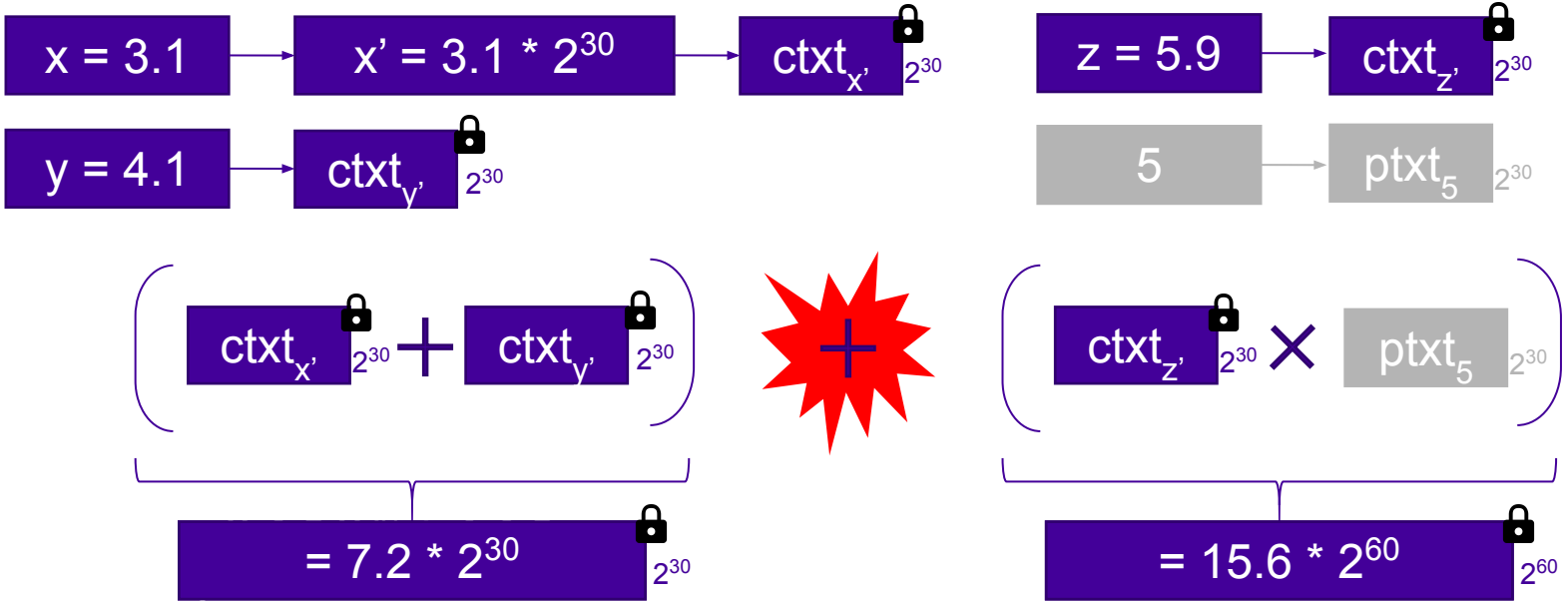
DEMO Time!

4

4.1. DEMO I: Client-Server interaction for encrypted integer operation



4.2. DEMO II: Teaching common CKKS pitfalls



Lab 13: FHE: (Ab)using the CKKS Scheme

Conclusion

5

5. Takeaways

- **PYFHEL**: Efficient Python wrapper for FHE libraries (***SEAL*** ☺, *PALISADE* [WIP])
  - One-click compilation & installation
  - Operator overloads & Python grammar
  - Access to underlying polynomials
- Nice tool for **implementations**, but also for **teaching**
- Next Steps: Extend to other FHE Libraries, unified API across libraries.

Try it out now:

```
pip install Pyfhel==3.0.0b1
```

CONTACT

Alberto IBARRONDO

PhD student at IDEMIA & EURECOM

ibarrond@eurecom.fr

Alexander VIAND

PhD student at ETH Zurich

alexander.viand@inf.ethz.ch