RAMPARTS

Rapid Machine-learning Processing Applications and Reconfigurable Targeting of Security

An IARPA-funded seedling exploring ease of use and feasibility of automatic correctness, optimization, and security settings for Fully Homomorphic Encryption

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.
RAMPARTS Team

Dr. Dave Archer, PI
Dr. Jose Calderon
Dr. Alex Malozemoff
Jason Dagit

Prof. Kurt Rohloff, PI
Ledah Casburn
Yuriy Polyakov
Jerry Ryan
Example FHE Use Case

Homomorphic Computation Host

Goals:
- Data remains private
- Only owner sees result
- Easy to program

Data Owner

Program F()

\( \text{Enc}_{\text{pubkey}}(\text{F}(\text{data})) \)

Public key \( \text{pubkey} \)

Data Warehouse

\( \text{Enc}_{\text{pubkey}}(\text{data}) \)
Problem: FHE Today is Hard to Use

Today: design by hand-tuning and expert knowledge

Challenge: Make FHE easier to use by engineers
Simple Metrics for Ease of Use

- **Expressive power**
  - How much of data analysts’ higher level language is supported?

- **Ease of programming**
  - How much do data analysts need to know to use encrypted computing capabilities?

- **Ease of deployment**
  - How easy to setup, install, deploy, re-use and maintain?

- **Ease of data access**
  - How easy is it to encrypt data for FHE use, and how much does it cost to transmit that data to the computation host?
Already substantial improvement on FHE ease of use, but
Only one-liners supported: Limited expressive power

```haskell
# declare our function, marking with the @fhe macro
# so that the compiler knows that it may be run under FHE
fp = Fhe.@fhe function f(db, pk, sk, a, b, c, d)
   ((a - b) * (a - b)) + ((c - d) * (c - d))
end

# Call the function, currently we have to use dummy data for
# values, this will be improved in future versions
println("Evaluating function on dataset")
res = fp("db", "owner.pk", "owner.sk", 0, 0, 0, 0)
println("Result (FHE): ", res)
```

Used modified Julia compiler

Limited ease of deployment
February 2017: a Better “Front End”

- Greater function support*:
  Better expressive power

- Simple interface:
  Better ease of programming

- Better ease of deployment

- Uses standard Julia compiler

*: currently in MATLAB, Julia by end of Phase II
November 2016: the “Back end”

- Ease of Programming
  - Integrated NULL and FV scheme support for arithmetic operations
- Ease of Deployment
  - Semi-automated configuration of FHE protocols
  - Wrapper for high-level usage
February 2017: A Better “Back End”

- **Ease of Data Access**
  - More data types supported
    - Ex: Matrices and Rationals

- **Ease of Programming**
  - Support for custom optimized FHE circuits.
    - Ex: Vectorized operations
    - Ex: Linear regression
  - Circuit estimator framework
    - Functional and performance estimator
Goal: From Julia Function to FHE

julia → Symbolic simulation → Circuit representation → PALISADE
Why Symbolic Execution?

- FHE uses circuits statically configured *before* execution
- (Imperative) programs dynamically configured *during* execution

To cross evaluation gap, use *symbolic execution*

- **Interpret** (almost) all execution paths in the program
- **Express** program values symbolically rather than concretely
- **Encode** terminal expressions for values as logic or arithmetic circuits

\[
return a*a + b*b
\]
Demonstrations, February 2017

Linear Regression
- Implemented in Julia/MATLAB
  - For functions written in “native” language without expertise
- Implemented in PALISADE
  - For functions written in “assembly” language to incorporate expert knowledge

Circuit Estimation
- Shows ability to estimate FHE circuit properties for compiler use
Demonstration: Linear Regression

- For demo, use MATLAB (vs. Julia) as language for compiling circuit
  - Pre-existing capabilities for MATLAB to symbolic simulator input
- Every other step in Julia

![Diagram](image.png)
Conclusions

- RAMPARTS focuses on ease of use and efficiency for practical FHE
  - Expressive power that’s easy to program
  - Simplicity of deployment
  - Simplicity of accessing data sets
  - Competent optimization of functions in FHE

- RAMPARTS prototypes are making novel contributions and progress in all of these areas
  - Symbolic execution generates FHE-ready circuits from analytic programs
  - Integrated FHE capability makes user experience match analyst expectations
  - Extensible FHE library allows for continuous enhancement of performance and capability

- RAMPARTS may offer a useful model for standardization of MPC interfaces beyond FHE alone