Parallelization of Fully Homomorphic Data Encoding

Jess Woods[1], Ada Sedova[2], Oscar Hernandez[1]

[1]*Computer Science Research Group, Oak Ridge National Laboratory* [2]*Biophysics Group, Oak Ridge National Laboratory*

Introduction

- Data privacy is important for healthcare data, secure computation, etc.
- *Homomorphic* encoding scheme: supports operations on encoded data (*Fig 1*)

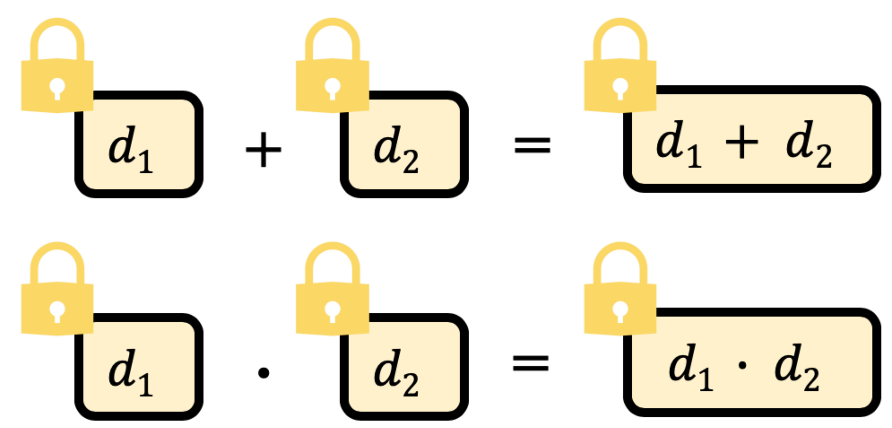

Figure 1: Homomorphic Operations

- *Fully homomorphic* scheme: supports arbitrary number of additions *and* multiplications [1]

Objective: Implement FHE scheme to allow user to outsource computation (the function f) to some "cloud" (separate user, supercomputer, etc.) without revealing any data (*Fig 2*)

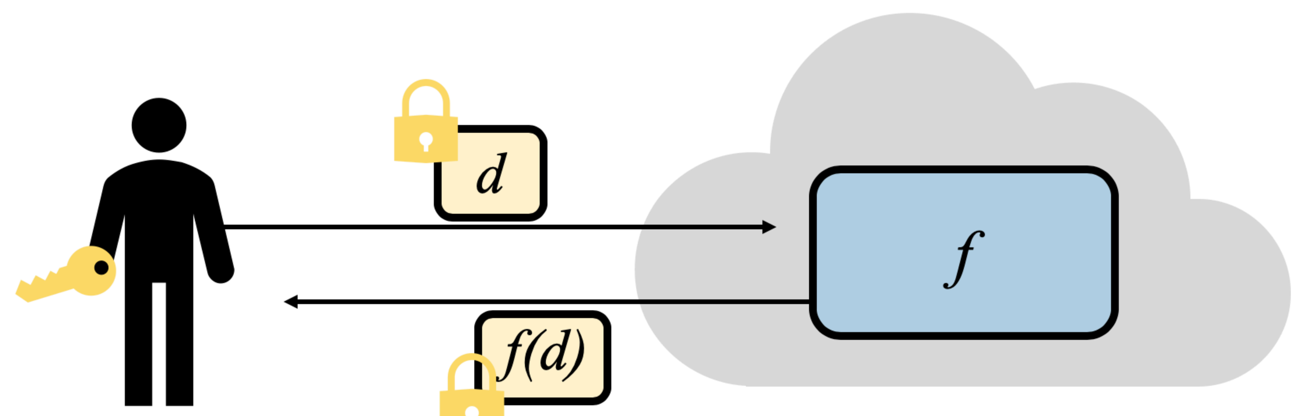

Figure 2: Desired Model

Problems: time and memory

- Costly large integers used for security
- Costly "Recode" operation, used to mitigate noise growth (red in *Fig 3*)

Hypothesis: A parallelized implementation will mitigate data transformation costs and make fully homomorphic encoding feasible

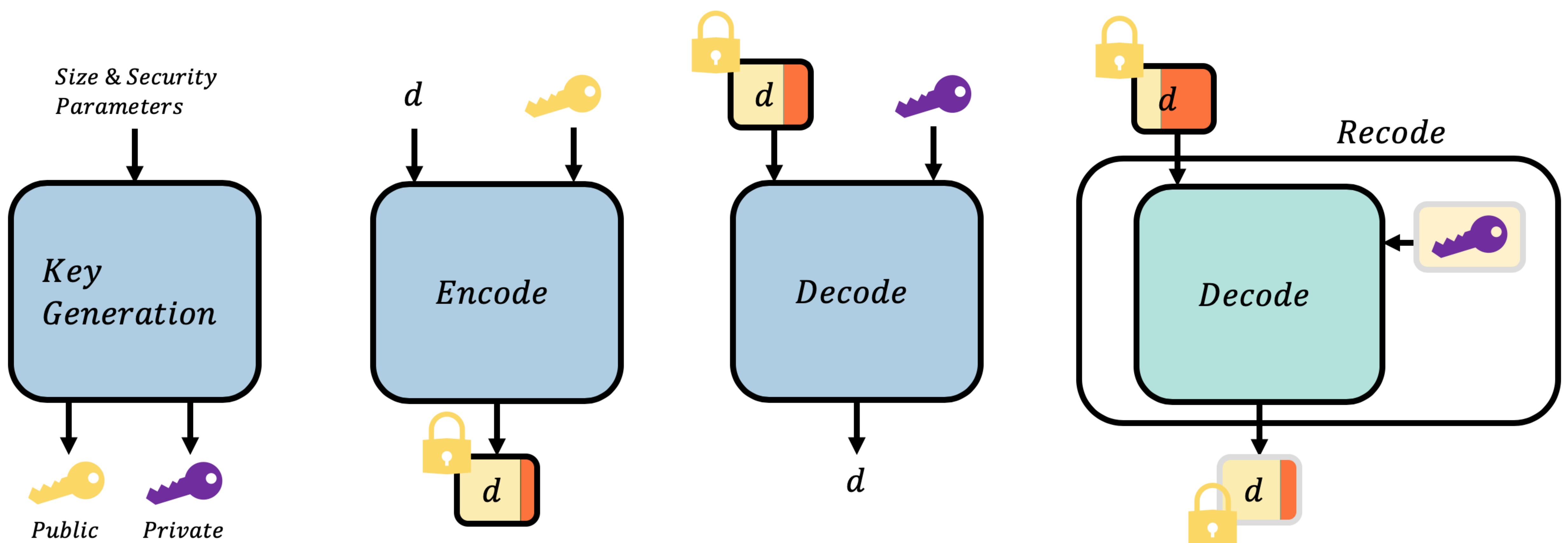

Figure 3: Scheme Operations

Methods

Scheme used: Dijk *et al.*'s fully homomorphic encoding scheme over the integers [2]

- Operations listed in *Fig 3*

Theoretical Improvements:

- Compressed public key size [3]
- Batched data (multiple bits per encoding) [4]

Implementation Improvements:

- GPU operations with CUDA
- Algorithm-level and OpenMP thread-level parallelism
- Big number handling with GMP library

Programming Languages:

- Python (proof of correctness sketch)
- C++ (better memory control)
- Julia (possible future of parallel computing)

Tested for:

- Summit (and future) supercomputers
- Smaller computing clusters
- Personal Laptop

Results

Our Contributions:

- First parallelization of Dijk *et al.*'s scheme
- First to incorporate both the theoretical improvements
- Current fastest implementation of Dijk scheme
- Our comparison of languages/models used to help design and choose future supercomputer software

Conclusion

Future applications:

- Secure machine learning, achieved with homomorphic matrix multiplications
- Healthcare transactions
- Library functions that hide complex operations

With such a library and our achieved speed-ups, fully homomorphic encoding will be more feasible than ever for practical use by non-specialists.

OAK RIDGE National Laboratory | OAK RIDGE INSTITUTE FOR SCIENCE AND EDUCATION | U.S. DEPARTMENT OF ENERGY | Office of Science

Acknowledgements: This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship program.

Contact: woodsjk@ornl.gov

[1] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.

[2] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[3] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–464. Springer, 2012.

[4] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 315–335. Springer, 2013.