Intel© Homomorphic Encryption Acceleration Library for FPGAs

Yan Meng,[1*] Fillipe D M de Souza,[1] Hubert de Lassus,[1] Flávio Bergarmaschi[2*]

[1]Intel Corporation, Folsom, CA USA; [2]Intel Corporation, UK

[*]To whom correspondence should be addressed; E-mail: he_fpga_support@intel.com.

Abstract

The protection of sensitive data in transit and at rest can be accomplished with traditional data encryption techniques. In this manner, the confidentiality, security and privacy of the data are preserved throughout its life cycle. This strategy requires the data to remain permanently encrypted. With the increasing need to obtain meaningful insights from sensitive data, such as in medical and financial records, so as to generate social and business value, the data must be decrypted for processing. During processing, the data becomes vulnerable to cyberattacks. This violates the protection of the data at all times. In 2009, the seminal work of Gentry gave birth to Fully Homomorphic Encryption (FHE), a technique that allows processing on encrypted data. In particular, FHE enables computation of arbitrary functions on encrypted data without ever decrypting it. The benefits of FHE come at the cost of high computational demand. The performance of algorithms evaluating functions on encrypted data is many orders of magnitude lower than their counterparts computing on unencrypted data. Narrowing this performance gap is key to unlock the capability of data privacy protection during computation, pervasively. We propose FPGAs compute architectures that can be used to obtain throughput performance gains in critical Homomorphic Encryption (HE) operations.

Privacy-Preserving Computing

The concept of privacy-preserving computing arises when two or more organizations wish to benefit from a collaboration that involves sharing and processing of private data whose ownership do not intersect. Fraud detection and more accurate medical diagnosis are common applications that could benefit from collaborative work involving private data. In practice, this is often impeded for competitive reasons and legal regulation requirements. Homomorphic Encryption (HE) is an encryption technique that can address this problem.

Homomorphic Encryption

Homomorphic Encryption (HE) is a type of encryption technique that enables computation whilst data remains encrypted (*1*). This capability has the potential to unlock a plethora of use cases to run on untrusted cloud computing infrastructures without violating data privacy and confidentiality requirements. Over the years, many types of HE schemes and HE software libraries implementing these schemes have been proposed. The HE software libraries provide APIs for application developers building privacy-preserving workloads. Microsoft SEAL (*2*), HELib (*3*) and PALISADE (*4*) are among the most popular open-source HE software libraries. They differ in implementation, performance and support of the types of HE schemes. However, they all carry something in common. Under the hood, their code implements algorithms that perform heavy modular polynomial arithmetic operations on large amounts of data. These operations and the size of the encrypted data (ciphertexts) form the basis of the compute bottlenecks for HE-based workloads.

Vision Statement

HE technology has the potential to transform the way protection of data privacy and confidentiality is handled in the digital data domain. With Intel HEXL for FPGA, Intel wishes to contribute to the development of a hardware and software ecosystem that supports deployment of HE-driven applications at large scale. We welcome collaborative efforts with both the academic and industry communities. To this end, we publish a preliminary work, under an open-source license, that attempts to bring up FPGAs as co-processors to improve throughput of HE opera-

tions. With this initiative, we invite both communities to design and co-engineer the future of privacy-preserving technology using FPGAs with us.

Intel HEXL for FPGA Open-Source Release

Similarly to the work in (*5*), we have developed FPGA kernels that implement common homomorphic encryption computing building blocks. Those kernels are provided under an open-source license and accessible at runtime through an open-source C/C++ FPGA Runtime API. The kernels are written in OpenCL 2.0 and allow HE kernel developers to offload homomorphic encryption operations to Intel FPGAs; in particular, the Stratix 10 GX 2800 (available as the Intel PAC D5005 product). These operations include the forward and inverse number-theoretic transform (NTT) and NTT-based modular polynomial multiplication (as a ciphertext multiplication). Even though FPGAs as accelerators typically constrain those operations to an I/O-bound regime, we believe these low-level primitives allow throughput performance gains via batched lazy executions.

Release Contents

- OpenCL-based FPGA processing elements architectures (kernels) for computationally costly building blocks of homomorphic encryption, including (modular) polynomial multiplication, forward and inverse number-theoretic transform (NTT) algorithms;
- C/C++ API for host runtime access of the FPGA processing elements that implement HE computing building blocks;
- Source codes under an open-source license to encourage contributions from both the academic and industry communities.

To learn more, refer to `https://github.com/intel/hexl-fpga/docs/paper/release.pdf`.

Contributions and Future Work

Future directions include the distribution of all the currently provided kernels ported to Intel oneAPI (*6, 7*), a unified programming paradigm for heterogeneous computing, and additional functionalities to support more homomorphic encryption operations, such as relinearization and rotation. The community of HE practinioners and researchers, from academia to industry, is invited and welcomed to contribute to this effort with new optimizations, additional FPGA kernels, and extending the C/C++ FPGA Runtime API. Learn more on how to contribute at CONTRIBUTING.md.

References

1. C. Gentry, *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009), pp. 169–178.

2. Microsoft SEAL (release 3.7), `https://github.com/Microsoft/SEAL` (2021). Microsoft Research, Redmond, WA.

3. S. Halevi, V. Shoup, Design and implementation of helib: a homomorphic encryption library, Cryptology ePrint Archive, Report 2020/1481 (2020). `https://ia.cr/2020/1481`.

4. PALISADE Lattice Cryptography Library (release 1.11.2), `https://palisade-crypto.org/` (2021).

5. F. Boemer, *et al.*, Intel HEXL (release 1.2), `https://github.com/intel/hexl` (2021).

6. Intel oneAPI a new era of heterogeneous computing, `https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html`. Accessed: 2021-11-28.

7. Intel oneAPI toolkits, `https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html`. Accessed: 2021-11-28.