Intel® Homomorphic Encryption Toolkit V1.1.0

1. Overview

The Intel Homomorphic Encryption (HE) toolkit is designed to make it fast and easy to evaluate homomorphic encryption technology on the 3rd Generation Intel® Xeon® Scalable Processors using libraries optimized to take advantage of the newest Intel hardware features. Additionally, the Intel HE-Toolkit is a great starting point for people new to homomorphic encryption, offering numerous sample kernels showing multiple examples of how the libraries can be used to implement common mathematical operations using SEAL or PALISADE. In addition, there are example applications which demonstrate how HE technology can be used to create secure applications.

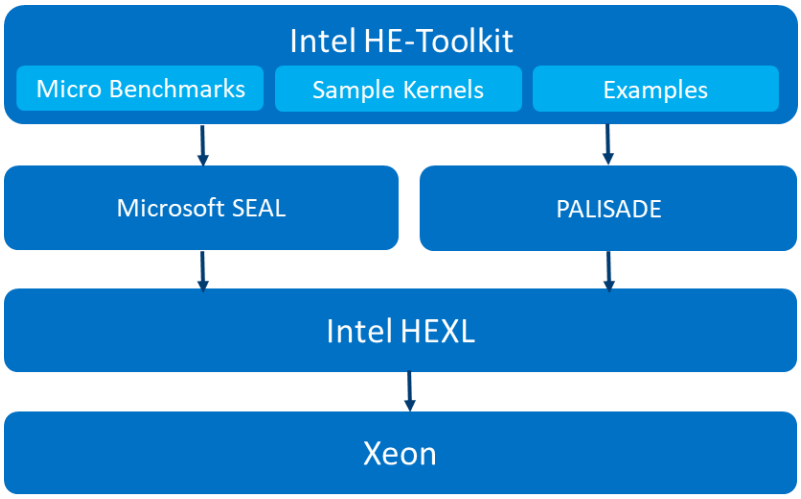

Figure 1 Intel optimized software stack

2. Intel HE Toolkit Components

The Intel HE Toolkit consists of several software components described in the following table.

Intel Homomorphic Encryption Acceleration library (HEXL)	HEXL offers optimized implementations utilizing AVX512 for some common mathematical operations performed during homomorphic encryption calculations.
Microsoft SEAL	Microsoft SEAL has the option to enable use of HEXL for supported HE calculations.
PALISADE	PALISADE has the option to enable use of HEXL for supported HE calculations

HE-Samples	• Micro benchmarks
	  ○ A set of benchmarks for basic homomorphic encryption operations built using Google Test.
	• Sample Kernels
	  ○ A set of sample kernels built using Google Test that show how higher level operations such as matrix multiplication can be implemented homomorphically.
	• HE examples
	  ○ A set of example applications showing how HE libraries can be used to enable different use cases.

3. Intel Homomorphic Encryption Acceleration library (HEXL)

Intel HEXL (https://github.com/intel/hexl) is designed to offer AVX512-optimized implementations for math operations commonly used in lattice-based homomorphic encryption libraries. The library currently offers optimized implementations for the following operations:

- number theoretic transform (NTT)
- element-wise vector-vector modular multiplication
- vector-scalar modular multiplication.

4. Microsoft SEAL

Microsoft SEAL now supports the option to enable the use of Intel HEXL to accelerate many Microsoft SEAL operations for the BFV and CKKS schemes. Support for HEXL can be enabled by compiling SEAL with the SEAL_USE_INTEL_HEXL flag set.

The version of Microsoft SEAL installed by the Intel HE Tool kit is based on

SEAL Git tag: v3.6.5

We currently see the largest performance benefit on the following operations when executing on a 3rd Generation Intel® Xeon® Scalable Processor.

Scheme	Operation
CKKS	Encrypt, Encode, Decrypt, Decode, Add, Multiply, Relin, Rescale, Rotate
BFV	Encrypt, Decrypt, Add, Multiply, Relin, Rotate

5. PALISADE

PALISADE now supports the option to enable the use of Intel HEXL to accelerate many PALISADE operations for the BFV and CKKS schemes. Support for HEXL can be enabled by compiling PALISADE with the WITH_INTEL_HEXL flag set.

The version of PALISADE installed by the Intel Toolkit is based on

PALISADE Git tag: v1.11.3

We currently see the largest performance benefit on the following operations when executing on a 3rd Generation Intel® Xeon® Scalable Processor.

Scheme	Operation
CKKS	Encode, Encrypt, Multiply, MultiplyRelinRescale, Rotate
BFV	Encode, Encrypt, Decrypt, Decode, Multiply, Rotate

6. HE Samples

HE Samples is a collection of software components built on Microsoft SEAL and PALISADE comprising of a set of micro-benchmarks, sample kernels for operations performed homomorphically, and example applications. The HE Samples are designed to enable quicker evaluation of HE on intel platforms, serve as a learning tool for how to

implement operations in different HE libraries, and provide examples of how these operations can be used to build applications based on HE technology for different use cases.

6.1 Micro-Benchmarks

HE Samples includes a set of micro-benchmarks for both Microsoft SEAL and PALISADE. These benchmarks are implemented using the google test framework and are focused on testing and benchmarking operations which are limited to one, or in some cases two API calls.

6.2 Sample Kernels

The sample kernels are implemented using google test for Microsoft SEAL and PALISADE. These sample kernels are for operations which are more advanced than the micro-benchmarks but are not full standalone use cases. An example of such an operation is matrix multiplication for which we provide several different sample kernels showing alternative methods of implementing the same algorithm using different approaches, libraries, and schemes. The full set of available sample kernels for each library and a brief description are described in the following tables. All sample kernels use N = 8192 with 3 Coefficient modulus primes.

Microsoft SEAL Sample Kernels	
Sample Kernel Name	Brief Description
DotPlainBatchAxis_CKKS	Ciphertext-plaintext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ CKKS.
DotCipherBatchAxis_CKKS	Ciphertext-ciphertext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ CKKS.
MatMulVal_CKKS	Matrix multiplication w/ matrix represented one row per ciphertext and manual computation using a collection of adds, multiplications, and rotations w/ CKKS.
LogisticRegression_CKKS	Computes logistic regression inference using a batch of inputs.
DotPlainBatchAxis_BFV	Ciphertext-plaintext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ BFV.
DotCipherBatchAxis_BFV	Ciphertext-ciphertext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ BFV.
MatMulVal_BFV	Matrix multiplication w/ matrix represented one row per ciphertext and manual computation using a collection of adds, multiplications, and rotations w/ BFV.
MatMulRow_BFV	Matrix multiplication w/ one matrix represented fully in a single ciphertext and the other matrix represented two rows at a time w/ BFV.

PALISADE Sample Kernels	
Sample Kernel Name	Brief Description
DotPlainBatchAxis_CKKS	Ciphertext-plaintext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ CKKS.
DotCipherBatchAxis_CKKS	Ciphertext-ciphertext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ CKKS.
MatMulEIP_CKKS	Matrix multiplication w/ matrix represented one row per ciphertext using Palisade's internal dot product calculation w/ CKKS.
MatMulVal_CKKS	Matrix multiplication w/ matrix represented one row per ciphertext and manual computation using a collection of adds, multiplications, and rotations w/ CKKS.
MatMulRow_CKKS	Matrix multiplication w/ one matrix represented fully in a single ciphertext and the other matrix represented row by row w/ CKKS.
LogisticRegression_CKKS	Computes logistic regression inference using a batch of inputs.
DotPlainBatchAxis_BFV	Ciphertext-plaintext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ BFV.
DotCipherBatchAxis_BFV	Ciphertext-ciphertext matrix multiplication w/ matrix represented as one element per ciphertext using a collection of adds and multiplications w/ BFV.
MatMulEIP_BFV	Matrix multiplication w/ matrix represented one row per ciphertext using Palisade's internal dot product calculation w/ BFV.
MatMulVal_BFV	Matrix multiplication w/ matrix represented one row per ciphertext and manual computation using a collection of adds, multiplications, and rotations w/ BFV.
MatMulRow_BFV	Matrix multiplication w/ one matrix represented fully in a single ciphertext and the other matrix represented row by row w/ BFV.

6.3 HE Examples

HE examples includes example applications built using HE technology. The primary purpose of these examples is to serve as a showcase of different use cases which can be implemented using HE, as well as learning references and starting points for further development. The toolkit currently includes one example application for secure query.

6.3.1 Secure Query

The secure query example shows how it is possible to implement a simple key, value database using HE and then allow a client to perform look ups of values in the database without exposing the query to the server hosting the database and optionally the key, value pairs in the database as well. The secure query example is implemented using the SEAL BFV scheme.

7. Performance benefits with 3rd Generation Intel® Xeon® Scalable Processors

This section highlights the performance benefits of Intel HEXL 1.1.0 on 3rd Generation Intel® Xeon® Scalable Processors. We provide reference performance improvements with HEXL enabled for Microsoft SEAL and PALISADE compared to the respective library with HEXL disabled.

In the following sections we provide a subset of performance data from our micro-benchmark test applications, which are available in the HE Samples micro benchmarks offered in the Intel HE Toolkit. The tests were performed on 5/28/2021.

7.1 Hardware Configuration

Table 1 shows the hardware configuration used for capturing performance data which is presented in section 7.3.

	HW Config (Valid for both config 1 & config 2)
# Nodes	2
# Sockets	2
CPU	Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz
Cores/socket, Threads/socket	36, 72
Microcode	0xd000270
HT	Enabled
Turbo	Enabled
Power management (disabled/enabled)	Enabled (Performance)
# NUMA nodes per socket (1, 2, 4...)	1
Prefetcher'e enabled (svr_info)	DCU HW, DCU IP, L2 HW, L2 Adj.
BIOS version	WLYDCRB1.SYS.0020.P93.2103190412
System DDR Mem Config: slots / cap / speed	Soc0: (Slots: 0, 4) (16GB @3200) Soc1: (Slots 0, 4) (16GB @3200)
Total Memory/Node (DDR, DCPMM)	64, 0
Storage - boot	SanDisk_SDSSA-1 (1 TB)
NIC	I210 Gigabit NIC

Table 1 System hardware configuration

7.2 Software configuration

Table 2 describes the software configuration used for capturing the performance results presented in section 7.3. The configurations differ by the inclusion of Intel HEXL and compiling Microsoft SEAL and PALISADE with HEXL enabled.

	Config 1 (baseline)	Config 2 (with Intel HEXL)
OS	Ubuntu 20.04.1 LTS	Ubuntu 20.04.1 LTS
Kernel	5.4.0-72-generic	5.4.0-72-generic
Compiler	Ubuntu clang version 10.0.1	Ubuntu clang version 10.0.1
Libraries	HE-Toolkit v1.1.0 PALISADE v.1.11.3 Microsoft SEAL v3.6.5	HE-Toolkit v1.1.0 PALISADE v.1.11.3 Microsoft SEAL v3.6.5 Intel HEXL v1.1.0
Other SW	CMake 3.19.2	CMake 3.19.2
Other SW	ldd (Ubuntu GLIBC 2.31-0ubuntu9) 2.31	ldd (Ubuntu GLIBC 2.31-0ubuntu9) 2.31

Table 2 Software configurations

7.3 Micro-kernel performance

Figure 2 shows the results of running the micro benchmarks for SEAL on 1-thread on the system described in table 1 using the software configurations described in table 2.

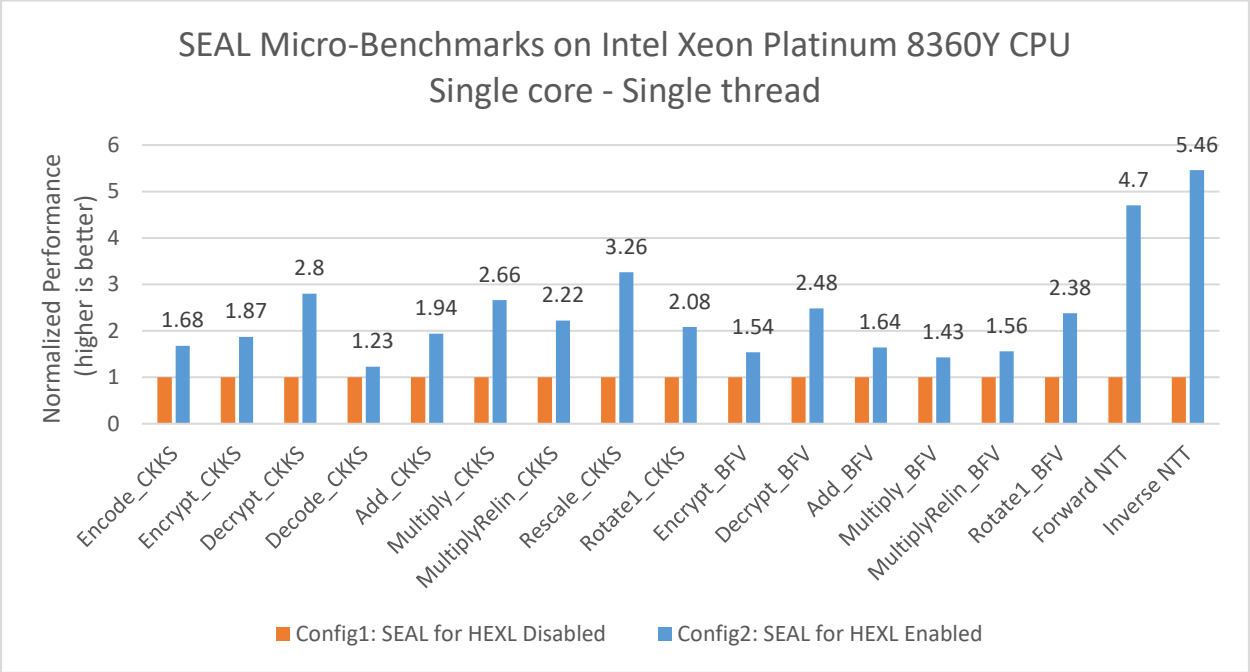

Figure 2 SEAL Micro-Benchmarks

Figure 3 shows the results of running the micro benchmarks for PALISADE on 1-thread on the system described in table 1 using the software configurations described in table 2.

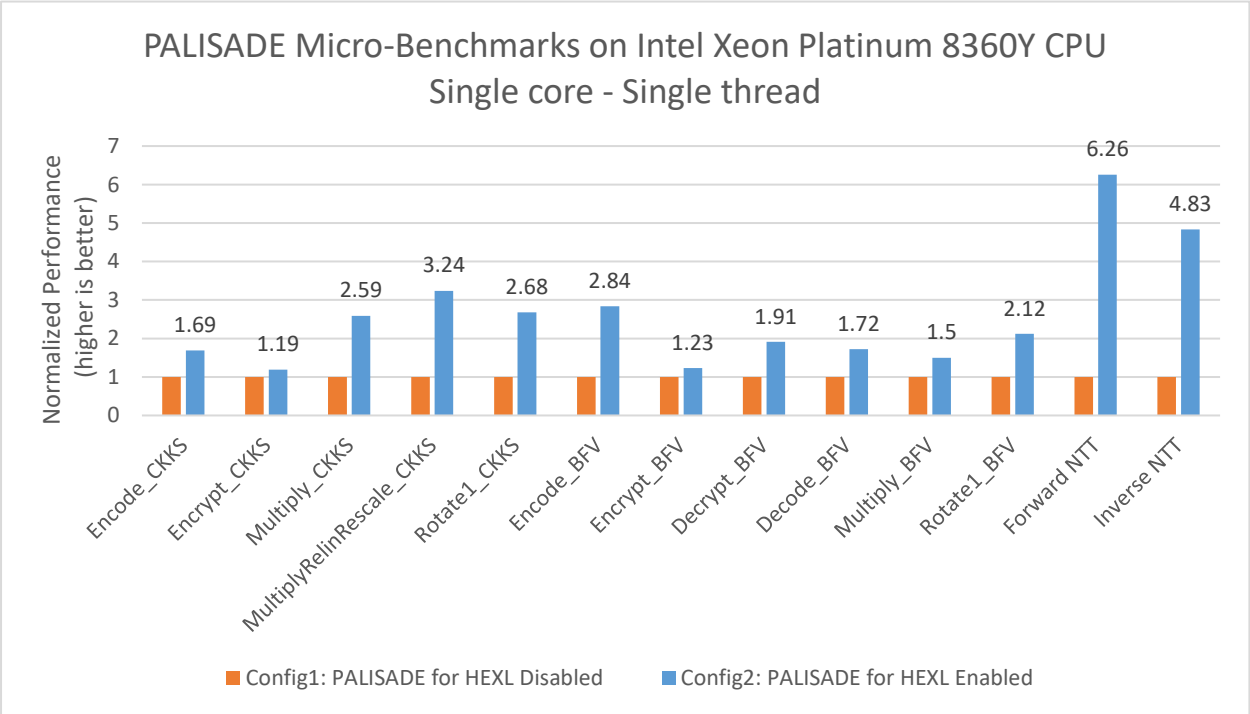

Figure 3 PALISADE Micro-benchmarks

8. Reference Documentation

PALISADE

SEAL

CKKS paper

BFV paper

Intel HEXL