Homomorphic Encryption Based Secure Sensor Data Processing

Vijay Gadepally*, Mihailo Isakov†, Rashmi Agrawal‡, Jeremy Kepner*, Karen Gettings*, Michel A. Kinsy†

*Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA
† Adaptive and Secure Computing Systems (ASCS) Laboratory
Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX
‡Department of Electrical and Computer Engineering, Boston University
Boston, MA

***Abstract*—Novel sensor processing algorithms face many hurdles to their adoption. Sensor processing environments have become increasingly difficult with an ever increasing array of threats. These threats have, in turn, raised the bar on deploying new capabilities. Many novel sensor processing algorithms exploit or induce randomness to boost algorithm performance. Co-designing this randomness with cryptographic features could be a powerful combination providing both improved algorithm performance and increased resiliency. The emerging field of signal processing in the encrypted domain has begun to explore such approaches. The development of this new class of algorithms will require new classes of tools. In particular, the foundational linear algebraic mathematics will need to be enhanced with cryptographic concepts to allow researchers to explore this new domain. This work highlights a relatively low overhead method that uses homomorphic encryption to enhance the resiliency of a part of a larger sensor processing pipeline.**

***Index Terms*—sensor data, homomorphic encryption, multi-party secure computing, matrix operations.**

I. Introduction

The increased use of small, networked sensors enabled by commercial-off-the-shelf (COTS) internet-of-things (IoT) technology has made sensor data processing an important computing challenge in a wide range of applications. The data processing algorithms implemented in these sensors range from simple, low-latency ones to complex, hard real-time, high-throughput ones. Advanced sensor processing algorithms such as compressed sensing [1], super-resolution [2], [3], [4], low-probability of detection (LPD) waveforms [5], bistatic sensors [6], [7], [8], data sampling [9], non-linear equalization [10], and multi-input-multiple-output (MIMO) [11], [12] techniques can be computationally intensive.

Nowadays, in sensor data processing applications, in addition to the conventional performance, latency and data storage challenges, serious issues of privacy and integrity are emerging. The privacy and integrity of both data and processing algorithm are vital in many safety or security critical applications [13]–[15]. The networked, distributed, and highly accessible nature of these sensors create a new and particularly challenging threat landscape [16]. The need to provide security and cyber-resiliency against these threats is yet another hurdle to the deployment of advanced sensor processing algorithms.

There are numerous techniques to protect sensitive data that have varying levels of computational overhead and varying levels of security [17]–[19]. For example, tools such as Computing on Masked Data (CMD) [20], [21], CryptDB [22] and Arx [23] use different types of encryption for different operations in order to minimize computational overhead (sometimes at the cost of security). In general, for a given application, there are a number of candidate privacy and integrity techniques that can be used. Each of these options typically have varying performance and security guarantees.

The emerging field of signal processing in the encrypted domain (SPED) [24] combines signal processing and encryption to make sensor data more resilient to classes of threats. A special issue of the IEEE Signal Processing Magazine discussed, in depth, the motivation for signal processing in the encrypted domain [24]. In this issue, various applications such as biometrics, media searching, and speech processing were discussed within the context of privacy preserving techniques such as Homomorphic Encryption (HE) [25] and Multiparty Computation (MPC) [26]. For example, in [27], the author discusses secure construction of a variety of signal processing and machine learning algorithms such as face recognition, clustering and recommender systems.

The above SPED examples provide resiliency to standard sensor processing algorithms. Advanced sensor processing algorithms present a wider range of opportunities to include SPED concepts. Many novel sensor processing algorithms exploit or induce randomness to boost algorithm performance. Co-designing this randomness with cryptographic features could be a powerful combination providing both improved algorithm performance and increased resiliency. In this work, we extract and examine two algebraic kernels that form the basis of HE and MPC algorithms within the context of matrix-vector operations that dominate sensor processing algorithms.

II. Linear Algebra Kernels and Operations

Commonly, signal processing algorithms express the operations to be performed on signals in a linear algebraic form

for compact and efficient execution. One of the fundamental building blocks of signal processing is convolution operation. Convolution is a operation that expresses the amount of overlap of a function as it is shifted over another function. Convolution can express the output of a linear time invariant system. Given an impulse response h and an input signal x, the output y is given as the convolution of x and h, denoted as $y = x * h$. These operations and many more can be expressed using linear algebra.

For example, let us consider a discrete signal $x = (x_1, x_2, ..., x_n)$ that is going through a system with impulse response $h = (h_1, h_2, ..., h_m)$. The result, $y = xh$ is given by:

$y_n = \sum_{m=-\infty}^{\infty} x_m h_{n-m}$.

This can also be written in matrix form as:

$$y = \begin{bmatrix} h_1 & 0 & ... & 0 \\ h_2 & h_1 & ... & 0 \\ ... & ... & ... & ... \\ h_{m-1} & ... & ... & h_1 \\ h_m & h_{m-1} & ... & h_2 \\ ... & ... & ... & ... \\ 0 & 0 & ... & h_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_{n-1} \\ x_n \end{bmatrix}$$

Essentially, the convolution becomes a matrix-vector multiplication. Now, suppose that we wish to keep either the signal or filter response encrypted. The following property allows us to do convolution (or other similar linear algebraic operations) between encrypted and non-encrypted signals when the encryption is done using a Paillier cryptosystem that supports homomorphic addition (where $E(m)$ signifies the encrypted version of the message/content/data m):

$$E(m_1 + m_2) = E(m_1) \ . \ E(m_2) \tag{1}$$

$$\prod_{i=1}^{c} E(m_1) = E(c \ . \ m_1) \tag{2}$$

Although, there are many signal processing algorithms with varying degrees of complexity, they are often using compositions of simple linear operations, which can be realized as a combination of Basic Linear Algebra Subprograms (BLAS) kernel calls [28]. Indeed, the BLAS kernels are used extensively in the linear algebra and signal processing community, including the emerging field of signal processing in the encrypted domain (SPED).

Operations in BLAS library can be classified in levels one through three as follows. The Level 1 BLAS - which is vector operations on strided array operation - is expressed as $y \leftarrow \alpha x + y$, where α and x are a scalar and a vector, respectively. The Level 2 BLAS expression - a generalized matrix-vector multiplication operation - is $y \leftarrow \alpha \mathbf{A} x + \beta y$. The level 3 BLAS expression — with matrix-matrix operations — is $y \leftarrow \alpha \mathbf{AB} x + \beta \mathbf{C}$.

III. Homomorphic Encryption Operation of Linear Algebra Kernels

In this section, we provide an overview on how different homomorphic encryption schemes can be used to perform the linear algebraic operations of the previous section.

A. Brief Overview of Homomorphic encryption from learning with errors

Gentry et al. [29] proposed a fully homomorphic encryption (FHE) scheme based on learning with errors (LWE) problem. This scheme is popularly known as the Gentry-Sahai-Waters (GSW) scheme and performs homomorphic encryption operations using approximate eigenvector method. The scheme is asymptotically faster as it computes matrix addition and multiplication for homomorphic addition and multiplication operations respectively. Other popular algorithms based on the hardness of the (Ring) Learning With Errors (RLWE) problem include Brakerski-Gentry-Vaikuntanathan (BGV) [30], Brakerski/Fan-Vercauteren (BFV) [31], and Cheon-Kim-Kim-Song (CKKS) [32]. There are open-source software libraries [33] and hardware primitives for accelerating the core of kernels in these different algorithms [34].

B. Secure Matrix Multiplication Using BFV/LWE-based Homomorphic Encryption

Doung et al. [35] proposed a way to securely multiply two matrices using the somewhat homomorphic encryption scheme BFV, based on RLWE. They advance the idea proposed by Yasuda et al. [36], [37] for secure inner product computation to perform matrix multiplication. They also propose a new method to pack a matrix into a single ciphertext so as to reduce the size of the ciphertext and also the computation cost to just one homomorphic multiplication to perform a matrix multiplication operation.

Observation

- A large integer can be packed in a single ciphertext to enable efficient computation of sums and products over packed ciphertexts. Hence, a message M of up to n bits is broken into a binary vector $(m_0, \ldots, m_{n-1})$ and associated with a polynomial of degree less than or equal to $n-1$ as follows:

$$pm(M) = \sum_{i=0}^{n-1} m_i x^i \tag{3}$$

- Using the packed ciphertexts, polynomial addition and multiplications can be performed in a straightforward way.
- To perform secure inner product using a single homomorphic multiplication, the packing can be done as follows:

$$pm^{(1)}(A) = \sum_{i=0}^{m-1} a_i x^i$$

$$pm^{(2)}(A) = -\sum_{i=0}^{m-1} a_i x^{n-i}$$

Synopsis

- Using the above packing method (to compute inner product) requires m^2 homomorphic multiplications to compute a matrix multiplication of $m \times m$ matrices. Hence, the authors propose two methods to reduce the number of computations.
- Binary matrix multiplication

  Let A and B be two $m \times m$ matrices with binary entries. Let $A_1, \ldots, A_m$ denote the row vectors of A and $B_1^T, \ldots, B_m^T$ the column vectors of B.

  In order to compute the matrix multiplication, AB of two binary matrices, one needs to compute the inner products $\langle A_i, B_j^T \rangle$ for $i, j = 1, \ldots, m$.

  Packing each row of $A_i = (a_{i,0}, \ldots, a_{i,m-1})$ and each column $B_j^T = (b_{j,0}, \ldots, b_{j,m-1})$ can be done as follows:

$$pm^{(1)}(A_i) = \sum_{u=0}^{m-1} a_{i,u} x^u$$

$$pm^{(2)}(B_j^T) = -\sum_{v=0}^{m-1} b_{j,v} x^{n-v}$$

  We can define following two polynomials using the above representation of matrix A and B:

$$Pol^{(1)}(A) = pm^{(1)}(A_1) + \cdots + pm^{(1)}(A_m) x^{m(m-1)}$$
$$Pol^{(2)}(B) = pm^{(2)}(B_1^T) + \cdots + pm^{(2)}(B_m^T) x^{m(m-1)}$$

  Now to obtain the matrix multiplication AB, there can be two different approaches.

  The **first approach** assumes $n \geq m^2$, then for each $j = 1, \ldots, m$, let $ct_j = ct_{mat}^{(1)}(A) * ct^{(2)}(B_j^T)$ and let $Dec(ct_j, sk) \in R_t$ denote the decryption result. Then, for each $j = 1, \ldots, m$, the inner product $\langle A_i, B_j^T \rangle$ is the coefficient of $x^{(i-1)m}$ in $Dec(ct_j, sk)$. This is true because $\text{Dec}(\text{ct}_j, sk) = Pol^{(1)}(A) \times pm^{(2)}(B_j^T)$ $= \sum_{i=1}^{m} pm^{(1)}(A_i) \times pm^{(2)}(B_j^T) x^{(i-1)m}$

  For a fixed index i, we have $pm^{(1)}(A_i) \times pm^{(2)}(B_j^T) x^{(i-1)m} = \langle A_i, B_j^T \rangle x^{(i-1)m} +$ other terms of degree $n - v + u + (i-1)m$, with $u \neq v$ and $u, v \in \{0, \ldots, m-1\}$. This implies that the terms of degree $(i-1)m$ is exactly the term of degree $(i-1)m$ in $pm^{(1)}(A_i) \times pm^{(2)}(B_j^T) x^{(i-1)m}$.

  With this approach, after m homomorphic multiplications over the Somewhat Homomorphic Encryption (SHE) scheme, we get the resulting matrix multiplication AB.

  The **second approach** assumes $n \geq m^3$, then for each $j = 1, \ldots, m$, let $ct_j = ct_{mat}^{(1)}(A) * ct_{mat}^{(2)}(B)$ and let $Dec(ct_j, sk) \in R_t$ denote the decryption result. Then, for each i and j, the inner product $\langle A_i, B_j^T \rangle$ is the coefficient of $x^{(j-1)m^2+(i-1)m}$ in $Dec(ct_j, sk)$. This is true because $\text{Dec}(\text{ct}_j, sk) = Pol^{(1)}(A) \times Pol^{(2)}(B)$ $= \sum_{i=1}^{m} \sum_{j=1}^{m} pm^{(1)}(A_i) \times pm^{(2)}(B_j^T) x^{(j-1)m^2+(i-1)m}$

  The term of degree $(j-1)m^2+(i-1)m$ in $Dec(ct, sk)$ is exactly the term of degree $(j-1)m^2 + (i-1)m$ in $pm^{(1)}(A_i) \times pm^{(2)}(B_j^T) x^{(j-1)m^2+(i-1)m} \in R$. Hence, the coefficient of $x^{(j-1)m^2+(i-1)m}$ in $Dec(ct, sk)$ gives the inner product $\langle A_i, B_j^T \rangle$.

  This approach requires only one homomorphic multiplication on packed ciphertexts for secure matrix multiplication.

- Non-binary matrix multiplication

  Let A and B be two $m \times m$ matrices whose entries are integers of less than p bits. Let $A^{(1)}, \ldots, A^{(m)}$ and $B^{(1)}, \ldots, B^{(m)}$ be the rows of A and B^T respectively. For $i = 0, \ldots, m-1$, we can write $A^{(i)} = (a_0^{(i)}, \ldots, a_{m-1}^{(i)})$ and $B^{(i)} = (b_0^{(i)}, \ldots, b_{m-1}^{(i)})$. For a chosen integer $r > 0$, each integral entry $a_k^{(i)}$ in the base-r representation can be written as follows:

$$a_k^{(i)} = \sum_{u=0}^{d-1} a_{ku}^{(i)} r^u \ with \ a_{ku}^{(i)} \in 0, 1, \ldots, r-1 \quad (4)$$

  where $d = \lceil log_r 2^p \rceil$.

  $a_k^{(i)}$ can be packed as

$$a_k^{(i)} = \sum_{u=0}^{d-1} a_{ku}^{(i)} x^u \in R = Z[x]/(x^n + 1) \quad (5)$$

  Then each row of $A^{(i)}$ and the column $B^{(j)}$ of A and B respectively, can be associated to the polynomials in the ring R as follows:

$$pm_{m,p,r}^{(1)}(A^{(i)}) = \sum_{k=0}^{m-1} a_k^{(i)}(x) x^{2kd} \quad (6)$$

$$pm_{m,p,r}^{(2)}(B^{(j)}) = -\sum_{l=0}^{m-1} b_l^{(j)}(x) x^{(n-2ld)} \quad (7)$$

  The final polynomials that can be associated to A and B are: $\text{Pol}^{(1)}(A) = pm_{m,p,r}^{(1)}(A^{(1)}) + \cdots + pm_{m,p,r}^{(1)}(A^{(m)}) x^{(m-1)2md} = \sum_{i=1}^{m} pm_{m,p,r}^{(1)}(A^{(i)}) x^{(i-1)2md}$ $\text{Pol}^{(2)}(B) = pm_{m,p,r}^{(2)}(B^{(1)}) + \cdots + pm_{m,p,r}^{(2)}(B^{(m)}) x^{(m-1)2m^2 d} = \sum_{j=1}^{m} pm_{m,p,r}^{(2)}(B^{(j)}) x^{(j-1)2m^2 d}$

  The **first approach** assumes $n \geq 2md(m+1)$, then for each $j = 1, \ldots, m$, let $ct_j = ct_{mat}(A) * ct_{m,p,r}^{(2)}(B^{(j)})$ and let $Dec(ct_j, sk) \in R_t$ denote the decryption result. Then, for $j = 1, \ldots, m$, the inner product $\langle A^{(i)}, B^{(j)} \rangle$ is the sum of the terms of degree greater or equal to $(i-2)2md$ and less than $(i-1)2md + 2d$ in $Dec(ct_j, sk)$ evaluated at $x = r$.

  The **second approach** assumes $n \geq 2m^3 d + 2md + 2d$, then for each $j = 1, \ldots, m$, let $ct = ct_{mat}^{(1)}(A) * ct_{mat}^{(2)}(B)$ and let $Dec(ct_j, sk) \in R_t$ denote the decryption result. Then, for each i and j, the inner

product $\langle A^{(i)}, B^{(j)} \rangle$ is the sum of the terms of degree greater or equal to $(i-2)2md + (j-1)2m^2d$ and less than $(i-1)2md + (j-1)2m^2d + 2d$ in $Dec(ct_j, sk)$ evaluated at $x = r$.

Pros

- The proposed approach can be used to perform a variety of other operations like matrix-vector operations, and distributed matrix operations. Vector additions and multiplications can be readily performed because of the underlying RLWE-scheme.

Cons

- For further operations, the computed coefficients will have to be rearranged. This cost can be huge with large matrices.
- The second approach in both the binary and non-binary matrix multiplication requires a large value of n making the approach very expensive.
- The proposed approach will work well for square matrices; rectangular matrix will have to be padded with zeros and converted to a square matrix before performing operations on them.

C. Generalized Matrix-Vector Operations Using RLWE-based Homomorphic Encryption

Halevi and Shoup [38] describe a way to perform some of the linear algebra operations like matrix-vector multiplication operations using HElib software library. HElib library implements homomorphic encryption proposed in the RLWE-based Brakerski-Gentry-Vaikuntanathan (BGV) scheme.

Synopsis:

- To perform matrix-vector multiplication, we need to perform $w = Av$ where w and v are vectors and we are given the columns of the matrix as vectors, $A = (c_0 \mid \ldots \mid c_{n-1})$. To multiply A and v, the operation that needs to be performed is $\sum_{i=0}^{n-1} v[i]c_i$.
- The vector v will need to be fully replicated to obtain the vectors $v_0, \cdots, v_{n-1}$, and then compute $w = \sum_{i=0}^{n-1} v_i \times c_i$.

  A full replication of vector $v = [0, 1, 2, 3]$ will yield $v_0 = [0, 0, 0, 0]$, $v_1 = [1, 1, 1, 1]$, $v_2 = [2, 2, 2, 2]$, $v_3 = [3, 3, 3, 3]$.
- If a matrix is presented in a row-order format instead, then we could transpose the matrix A and then use the similar procedure listed above to perform matrix-vector multiplication. Another approach could be to have A stored as vectors $r_0, \cdots, r_{n-1}$. First compute the vectors $p_i = v \times r_i$ for $i \in [n]$ and then compute the entries of w by computing $w[i] = \sum_i p_i[j]$ for $i \in [n]$.
- Another convenient representation of a matrix is the diagonal order representation (more useful if a matrix is to be kept in plaintext).

  We represent the matrix by n vectors $d_0, \cdots, d_{n-1}$ that contain the generalized diagonals of A, namely, $d_i = (A_{0,i}, A_{1,i+1}, \cdots, A_{n-1,n+i-1})$, so $d_i[j] = A_{j,j+1}$.

  Example of diagonal order representation of matrix:

$$\text{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

$$= \begin{bmatrix} a_{0,0} & a_{1,1} & a_{2,2} & a_{3,3} \\ a_{0,1} & a_{1,2} & a_{2,3} & a_{3,0} \\ a_{0,2} & a_{1,3} & a_{2,0} & a_{3,1} \\ a_{0,3} & a_{1,0} & a_{2,1} & a_{3,2} \end{bmatrix}$$

  Then, the product $w = Av$ can be computed as $w = \sum_{i=0}^{n-1} d_i \times (v \ll i)$, where $\ll$ represents rotation of v by i times.

  By rotating the vector v, aim is to achieve $[v_0, v_1, v_2, v_3], [v_1, v_2, v_3, v_0], [v_2, v_3, v_0, v_1], [v_3, v_0, v_1, v_2]$.

  This approach will require $n - 1$ rotations, and n multiplications and additions to perform a matrix-vector multiplication.
- Mishra et al., in their work [39], make use of the linear algebra operations proposed here along with their own previous work [35] and propose efficient ways to perform matrix multiplication operations.

Pros:

- The proposed approach can be easily extended to perform matrix-matrix operations.

Cons:

- The elements will require to be rearranged to its correct location after the operations are performed.

D. Paillier-Based Private Multi-party Matrix Multiplication and Trust Computations

Dumas et al. [40] propose a protocol for securely computing dot-product of two vectors wherein elements of the vector are distributed among n parties. The Protocol is based on Paillier cryptosystem which is secure in the semi-honest model or secure with high probability in the malicious adversary model. The authors then advance the protocol to perform distributed matrix multiplication.

Observation

- To define a multi-party computation protocol which allows to efficiently compute a distributed matrix product with data distributed between players, the problem can be reduced to the computation of a dot product between vectors U and V such that one player knows U and V is divided between players.

Synopsis:

- As per the Paillier cryptosystem, homomorphic addition operation is performed as follows:

$$D(E(m_1, r_1) \cdot E(m_2, r_2) mod\ n^2) = (m_1 + m_2) mod\ n \tag{8}$$

  The homomorphic multiplication, with encrypted m_1 and m_2 in cleartext can be performed as follows:

$$D(E(m_1, r_1)^{m_2} mod\ n^2) = (m_1 m_2) mod\ n \tag{9}$$

TABLE I
COMPLEXITY ANALYSIS OF THE DIFFERENT HOMOMORPIC ENCRYPTION TECHNIQUES AS RELATED TO THEIR ALGEBRAIC OPERATIONS

Reference	Scheme	Allowed HE Operations	Linear Algebra Operations	Complexity (No. of HE Operations)
[29]	LWE-based	Add, Mul	Matrix Add, Matrix Mul, Mult by Constant,	$O(1)$
[35]	RLWE-based	Add, Mul	Vector Inner Product, Matrix Add, Matrix Mul	$O(1)$, $O(m)$, $O(m)$ and $O(1)$
[40]	Paillier	Add, Mul	Vector Inner Product, Vector Addition, Matrix Add, Matrix Mul	$O(m)$, $O(m)$, $O(m^2)$, $O(m^2)$

An encrypted plaintext raised to a constant k will decrypt to the product of the plaintext and the constant,

$$D(E(m_1, r_1)^k mod\ n^2) = (km_1) mod\ n \qquad (10)$$

- A linear dot product protocol with three parties:
  Alice is interested in computing a one-dimension 3 dot-product $S = u^T.v$ between her vector u and a vector v whose coefficients are owned by different parties.

  The other parties send their coefficients, encrypted to Alice. Then she homomorphically multiplies each one of these by her u_i coefficients and masks the obtained $u_i v_i$ by a random value r_i.

  Then the other parties can decrypt the resulting $u_i v_i + r_i$ with two unknowns u_i and r_i; they are not able to recover v_i.

  Finally the parties enter a ring computation of the overall sum before sending it to Alice. Then only, Alice removes her random masks to recover the final dot-product. Since at least two parties have added $u_2 v_2 + u_3 v_3$, there is at least two unknowns for Alice, but a single equation.

  After several decryptions and re-encryptions, and removal of the random values r_i, S is exactly $\sum u_i v_i$
- This linear dot product protocol can be generalized to n parties and requires $O(n)$ communications and $O(n)$ encryption and decryption.
- To compute the matrix multiplication, using this dot-product protocol, each party P_i owns two rows, A_i and B_i, one of each $n \times n$ matrices A and B. In order to compute the matrix product, it is required to parallelize the dot-product protocol: each dot-product is cut into blocks of 2 or 3 coefficients. At the end, each P_i learns a row C_i of the matrix $C = AB$.

Pros:

- Matrix multiplication can be performed in $O(n^2)$ rather than $O(n^3)$.
- Communication cost is linear.

Cons:

- With Paillier cryptosystem, there is no way to multiply two encrypted vectors or matrices. Only one has to be encrypted and the other has to stay as plaintext.

IV. COMPLEXITY STUDY OF THE DIFFERENT ALGORITHMS

In this section we summarize the operations supported by the different algorithms. Each algorithm supports the three different levels of BLAS. The results are summarized in Table I.

In addition to the analytic evaluation, we provide a quantitative example that further highlights the practical complexities of these algorithms when use for BLAS operations. For this purpose, we use the BGV scheme of the HElib library and the Intel PIN Tool [41] for runtime instrumentation. The BGV variant in the HELib package is defined over polynomial rings of the form $= [X]/\Phi_m(X)$ where m is a parameter and $\Phi_m(X)$ is the m'th cyclotomic polynomial. The native plaintext space is $[X]/(\Phi_m(X), p^r)$ where r is a small value and p defines the modulo-p polynomials. The vector size used in this illustrative test case is a 64-integer entry vector, with $p = 4999$, $m = 32109$ and $r = 1$ (for the Level 1 scenario, we convert the matrix to a vector). Tests were performed on an Intel Core i7-5820K Processor with 6-Cores, 3.3GHz, 15MB Cache, with Hyper-Threading Technology machine. In all the runs, we use a single-threaded version of the program. The cycle numbers reported are gathered through instrumentation using the Intel PIN Tool. The cycle count is the end-to-end program run with all related system calls.

The key observations are (1) homomorphic encryption based computation remains an expensive task - even for simple matrix operations, and (2) creating the parameters and computing objects to set up the execution environment with the desirable level of security is the computationally intensive part of the operation. This part occupies 85% of the total execution cycles.

V. CONCLUSION AND FUTURE WORK

In this work, we highlight the importance of signal processing in the encrypted domain (SPED). We connect this growing application domain to conventional BLAS kernels. We then go through a full overview of key homomorphic encryption algorithms that lend themselves to performing these computations securely. Finally, we provide their time complexities and a computational analysis of one of these algorithms. In our future, we plan to select three representative signal processing applications (end-to-end), highlight their core linear algebra kernels, and how one can efficiently perform secure computation on them.

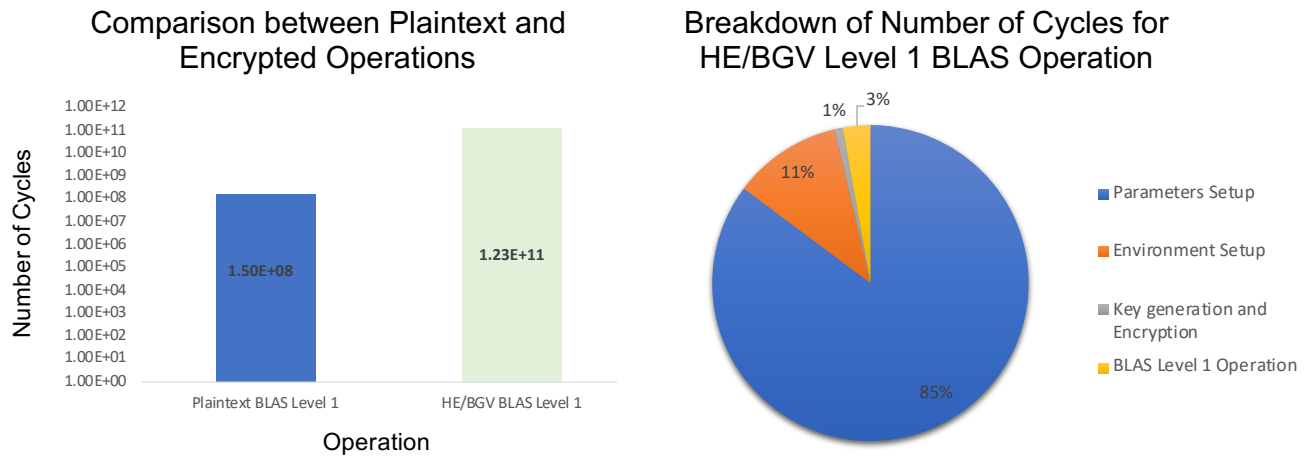

Fig. 1. BLAS Level 1 homomorphic encryption execution using the RLWE-based Brakerski-Gentry-Vaikuntanathan (BGV) scheme. The left side of the figure describes the computational overhead using the number of cycles. As observed in the figure, the overhead for the encrypted operation is approximately 10^3 cycles. On the right side of the figure, we show the breakdown of number of cycles as they are used for different portions of the encrypted operation. For "Environment Setup," this includes tasks such as building modulus chain and the switching matrices.

Acknowledgements

The authors wish to thank Paul Monticciolo, Albert Reuther, Emily Shen, Rob Cunningham, and Arkady Yerukhimovich for early discussions related to this work. The authors also wish to thank William Arcand, David Bestor, William Bergeron, Chansup Byun, Matthew Hubbell, Michael Houle, Michael Jones, Anna Klein, Peter Michaleas, Lauren Milechin, Julie Mullen, Andrew Prout, Antonio Rosa, Sid Samsi, and Charles Yee.

References

[1] B. Miller, J. I. Goodman, K. W. Forsythe, J. Z. Sun, and V. K. Goyal, "A multi-sensor compressed sensing receiver: Performance bounds and simulated results," *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pp. 1571–1575, 2009.

[2] I. G. Stiglitz, "The development of superresolution at Lincoln Laboratory," *MIT Lincoln Laboratory Journal*, vol. 10, no. 2, 1997.

[3] G. F. Hatke, "Superresolution source location with planar arrays," *MIT Lincoln Laboratory Journal*, vol. 10, no. 2, 1997.

[4] G. R. Benitz, "High-definition vector imaging," *MIT Lincoln Laboratory Journal*, vol. 10, no. 2, 1997.

[5] P. Pace, *Detecting and Classifying Low Probability of Intercept Radar*, ser. Artech House radar library. Artech House, 2009.

[6] C. Pearson and S. Pohlig, "Bistatic clutter suppression and target detection analysis for the RADARSAT/GMTI experiment," *Adaptive Sensor Array Processing Workshop (ASAP)*, 2002.

[7] J. Griesbach, "Bistatic radar clutter suppression error sensitivity," *Adaptive Sensor Array Processing Workshop (ASAP)*, 2002.

[8] G. R. Benitz, "A look at bistatic STAP from the viewpoint of SAR image formation," *Adaptive Sensor Array Processing Workshop (ASAP)*, 2002.

[9] V. Gadepally, T. Herr, L. Johnson, L. Milechin, M. Milosavljevic, and B. A. Miller, "Sampling operations on big data," in *2015 49th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2015, pp. 1515–1519.

[10] J. Goodman, B. Miller, M. Herman, G. Raz, and J. Jackson, "Polyphase nonlinear equalization of time-interleaved analog-to-digital converters," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 3, pp. 362–373, 2009.

[11] D. J. Rabideau and P. Parker, "Ubiquitous MIMO multifunction digital array radar," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 1, 2003, pp. 1057–1064 Vol.1.

[12] D. Bliss and K. Forsythe, "Multiple-input multiple-output (MIMO) radar and imaging: Degrees of freedom and resolution," in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, M. Matthews, Ed., vol. 1, 2003, pp. 54–59.

[13] M. Barni, G. Droandi, and R. Lazzeretti, "Privacy protection in biometric-based recognition systems: A marriage between cryptography and signal processing," *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 66–76, 2015.

[14] M. Gomez-Barrero, E. Maiorana, J. Galbally, P. Campisi, and J. Fierrez, "Multi-biometric template protection based on homomorphic encryption," *Pattern Recognition*, vol. 67, pp. 149 – 163, 2017.

[15] S. Yu, "Big privacy: Challenges and opportunities of privacy study in the age of big data," *IEEE Access*, vol. 4, pp. 2751–2763, 2016.

[16] M. Isakov, V. Gadepally, K. M. Gettings, and M. A. Kinsy, "Survey of attacks and defenses on edge-deployed neural networks," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019, pp. 1–8.

[17] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham, "Sok: Cryptographically protected database search," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 172–191.

[18] S. Yakoubov, V. Gadepally, N. Schear, E. Shen, and A. Yerukhimovich, "A survey of cryptographic approaches to securing big-data analytics in the cloud," in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2014, pp. 1–6.

[19] R. Agrawal, L. Bu, E. D. Rosario, and M. A. Kinsy, "Design-flow methodology for secure group anonymous authentication," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1544–1549.

[20] J. Kepner, V. Gadepally, P. Michaleas, N. Schear, M. Varia, A. Yerukhimovich, and R. K. Cunningham, "Computing on masked data: a high performance method for improving big data veracity," in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2014, pp. 1–6.

[21] V. Gadepally, B. Hancock, B. Kaiser, J. Kepner, P. Michaleas, M. Varia, and A. Yerukhimovich, "Computing on masked data to improve the security of big data," in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 2015, pp. 1–6.

[22] R. A. Popa, C. M. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 2011, pp. 85–100.

[23] R. Poddar, T. Boelter, and R. A. Popa, "Arx: an encrypted database using semantically secure encryption," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1664–1678, 2019.

[24] M. Barni, T. Kalker, and S. Katzenbeisser, "Inspiring new research in

the field of signal processing in the encrypted domain [from the guest editors],” *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 16–16, 2013.

[25] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–35, 2018.

[26] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure multiparty computation*. Cambridge University Press, 2015.

[27] Z. Erkin, “Secure signal processing: Privacy preserving cryptographic protocols for multimedia,” 2010.

[28] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, “A set of level 3 basic linear algebra subprograms,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 16, no. 1, pp. 1–17, 1990.

[29] C. Gentry, A. Sahai, and B. Waters, “Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based,” pp. 75–92, 2013.

[30] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “Fully homomorphic encryption without bootstrapping,” Cryptology ePrint Archive, Report 2011/277, 2011, https://eprint.iacr.org/2011/277.

[31] J. Fan and F. Vercauteren, “Somewhat practical fully homomorphic encryption,” Cryptology ePrint Archive, Report 2012/144, 2012.

[32] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” Cryptology ePrint Archive, Report 2016/421, 2016.

[33] “PALISADE Lattice Cryptography Library (release 1.9.2),” https://palisade-crypto.org/, 2020.

[34] R. Agrawal, L. Bu, and M. A. Kinsy, “Fast arithmetic hardware library for rlwe-based homomorphic encryption,” *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2020.

[35] D. H. Duong, P. K. Mishra, and M. Yasuda, “Efficient secure matrix multiplication over lwe-based homomorphic encryption,” *Tatra mountains mathematical publications*, vol. 67, no. 1, pp. 69–83, 2016.

[36] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba, “New packing method in somewhat homomorphic encryption and its applications,” *Security and Communication Networks*, vol. 8, no. 13, pp. 2194–2213, 2015.

[37] ——, “Secure statistical analysis using rlwe-based homomorphic encryption,” in *Australasian Conference on Information Security and Privacy*. Springer, 2015, pp. 471–487.

[38] S. Halevi and V. Shoup, “Algorithms in helib,” in *Advances in Cryptology – CRYPTO 2014*, J. A. Garay and R. Gennaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 554–571.

[39] P. K. Mishra, D. Rathee, D. H. Duong, and M. Yasuda, “Fast secure matrix multiplications over ring-based homomorphic encryption.” *IACR Cryptology ePrint Archive*, vol. 2018, p. 663, 2018.

[40] J.-G. Dumas, P. Lafourcade, J.-B. Orfila, and M. Puys, “Private multiparty matrix multiplication and trust computations,” *arXiv preprint arXiv:1607.03629*, 2016.

[41] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, “Pin: Building customized program analysis tools with dynamic instrumentation,” ser. PLDI ’05. New York, NY, USA: Association for Computing Machinery, 2005, p. 190–200.