Crypt-CNN(I): Secure Two-party Computation of Large-scale Matrix-vector Multiplication

Wen-jie Lu[1] Jun Sakuma[1,2,3]

Abstract: Matrix-vector multiplication is a fundamental computation of many machine learning algorithms, such as deep learning which involves large-scale matrices. However, practical secure two-party computations for large-scale matrix-vector multiplication are absent in the literature. The absence of practical approach for large-scale matrix-vector multiplication hinders the development of private evaluation of deep learning algorithms. We propose two practical approaches for the large-scale matrix-vector multiplication using fully homomorphic encryption under the secure two-party computation setting. Our approaches are efficient regarding computation time and bandwidth consumption. Our secure matrix-vector multiplication protocol can process a matrix with more than 4000×2000 elements within 8 seconds, which is about 3.6 times faster, and save about 93.8% bandwidth computation than the previous approach.

Keywords: PWS, matrix-vector multiplication, 2PC, Message Packing, Machine Learning

1. Introduction

Matrix-vector multiplication is a fundamental computation of many machine learning algorithms, such as deep learning which involves large-scale matrices. For instance, the state-of-the-art deep learning model [12] includes large-scale matrices up to 4096×100352 elements. In the plaintext domain, highly optimized libraries, such as BLAS and Eigen, enable us to process such large-scale matrix-vector multiplications efficiently.

Also, the concerns of data privacy motivate researchers to develop methods and protocols that privately evaluate machine learning algorithms without leaking the information of the data [1], [14], [15]. One approach for that is to use homomorphic encryptions, by which we can perform arithmetic operations above ciphertexts without knowing the plain messages.

Under the secure two-party computation setting [6], we can easily design a secure protocol for the matrix-vector multiplication using additively homomorphic encryption such as Paillier cryptosystem [10]. For instance, Alice encrypts elements of the matrix using the Paillier cryptosystem, and sends the ciphertexts to Bob. Then Bob can evaluate the matrix-vector multiplication with public-key operations of the Paillier cryptosystem. Unfortunately, such direct approach costs too much computation time or consume too many network bandwidth when the matrix is large. In this work, under the secure two-party computation setting, we propose two practical methods for large-scale matrix-vector multiplication. We show (in our second paper) that we can *evaluate* a 10-layer deep learning model using these two methods within one minute, which is about 10 times faster than the previous work [5].

Related Works. Yasuda et al. [17] proposed an computation efficient method for inner product of encrypted vectors. This method can be directly employed to the secure matrix-vector multiplication, by processing each row of the matrix separately. However, as we will show in Section 4.1, this direct approach is communication inefficient when the matrix contains too many number of rows. We empirically show that, when to process 4000 rows, this approach requires to transfer more than 4GB data, which is far from practical.

Our Contributions. We present two practical fully homomorphic encryption (FHE)-based building blocks for matrix–vector multiplication of these two types, separately. We show experimentally that our building blocks are 3.6 times faster in computation time and they reduce 93.8% bandwidth consumption, compared with methods described in past works.

2. Preliminaries

2.1 Notation

We begin with some related notations. For a positive integer n, we write $[n]$ to denote $\{0, 1, \ldots, n-1\}$. We write $e \xleftarrow{\$} \mathcal{E}$ to denote that e is chosen uniformly at random from the set $\mathcal{E}$. We denote vectors as bold lower case Roman characters, e.g., $\boldsymbol{v}$. We write $[v_0, v_1, \ldots]$ to explicitly denote the elements of a row vector. We designate matrices with bold upper case Roman characters, e.g., $\boldsymbol{M}$. We access elements of vectors with $(\cdot)_i$. For example, $(\boldsymbol{v})_i$ is the i-th element of $\boldsymbol{v}$. Similarly, tensors are shown with indices such

[1] University of Tsukuba
[2] JST CREST
[3] RIKEN center of AIP

as $(\boldsymbol{M})_{i,j}$. The concatenation of vectors is denoted as $\boldsymbol{a}\|\boldsymbol{b}$. $|\boldsymbol{v}|$ returns the length of $\boldsymbol{v}$. The transpose is written with $\cdot^{\top}$. The inner product of two column vectors is written as $\boldsymbol{u}^{\top}\boldsymbol{v}$; matrix–vector multiplication is denoted as $\boldsymbol{M}\boldsymbol{v}$. We write $\boldsymbol{M}_{[i:]}$ to denote the i-th row vector of the matrix $\boldsymbol{M}$ and write $\boldsymbol{M}_{[:j]}$ to denote the j-th column vector.

We use non-bold upper case Roman characters to denote polynomials. Specifically, symbol X is used to denote the indeterminate of polynomials. For an odd modulus t, $\mathbb{Z}_t$ is interpreted in the range $(-t/2, t/2)$. $\mathbb{Z}_t[X]$ denotes a set of polynomials whose coefficients are in $\mathbb{Z}_t$. We access coefficients of a polynomial P with $[\cdot]_i$. For example, the 1-st coefficient of the polynomial $P = 1 + 2X + 3X^2$ is given as $[P]_1 = 2$. Indices start from 0.

2.2 Matrix-vector Multiplication

We consider two kinds of matrix–vector multiplications. Specifically, for the first one, Alice's input is a long vector, and Bob's input is a large-scale matrix. Only one large-scale matrix-vector multiplication is needed for the evaluation. For the second one, Alice's input consists of small matrices, and Bob's input consists of same number of small vectors. A batch of small-scale matrix-vector multiplications must be used. Based on this subtle but significant difference, we design a secure 2PC protocol separately for these two kinds.

Put abstractly, our constructions are reduced to the secure 2PCs presented in Table 1. $\mathsf{MvM\text{-}v}$ signifies the matrix–vector multiplication with a vector as Alice's input. $\mathsf{MvM\text{-}m}$ denotes the matrix–vector multiplication with matrices as Alice's input. In this work, we present these 2PCs in a manner that the input of Alice is in a form of secret shares, and the evaluation result is also in a form of secret shares. This secret-share manner allows combinations of our protocols with other secure computation tools, such as Yao's garbled circuit [16]. We remark that our protocols are not designed for this secret-share manner specifically.

In both settings, we also assume Alice generates the (homomorphic) key pair $(\mathsf{sk}, \mathsf{pk})$ while Bob can only access to the public key pk. For the $\mathsf{MvM\text{-}v}$ (resp. $\mathsf{MvM\text{-}m}$) setting, Alice encrypts $\boldsymbol{v}+\boldsymbol{r}$ (resp. $\{\boldsymbol{M}_i+\boldsymbol{R}_i\}_i$) with FHE and sends the ciphertext(s) to Bob. The Bob performs homomorphic operations with the ciphertext(s) and its input $\{\boldsymbol{M}, \boldsymbol{r}\}$ (resp. $\{\boldsymbol{v}_i, \boldsymbol{R}_i\}_i$) to obtain the ciphertext of $\boldsymbol{M}\boldsymbol{v}$ (resp. $\sum_i \boldsymbol{M}_i\boldsymbol{v}_i$). After the computation, Bob homomorphically add its private share $\boldsymbol{r}'$ to the resulting ciphertext(s). Then, Bob sends the ciphertext(s) back to Alice. In the end, Alice learns its share $\boldsymbol{M}\boldsymbol{v} + \boldsymbol{r}'$ (resp. $\sum_i \boldsymbol{M}_i\boldsymbol{v}_i + \boldsymbol{r}'$) but nothing else. The Bob learns its private share $\boldsymbol{r}'$ but nothing else. We respectively designate these two functionalities as

$$\mathsf{MvM\text{-}v}(\boldsymbol{v}+\boldsymbol{r}, \{\boldsymbol{M}, \boldsymbol{r}\}) \rightarrow (\boldsymbol{M}\boldsymbol{v}+\boldsymbol{r}', \boldsymbol{r}'),$$
$$\mathsf{MvM\text{-}m}(\{\boldsymbol{M}_i+\boldsymbol{R}_i\}_i, \{\boldsymbol{v}_i, \boldsymbol{R}_i\}_i) \rightarrow (\sum_i \boldsymbol{M}_i\boldsymbol{v}_i+\boldsymbol{r}', \boldsymbol{r}').$$

We present our constructions of $\mathsf{MvM\text{-}v}$ and $\mathsf{MvM\text{-}m}$ in Section 4 using existing primitives introduced in Section 3.

Table 1 Matrix-vector multiplication of two kinds.

	Alice		Bob	
	Input	Output	Input	Output
MvM-v	$\boldsymbol{v}+\boldsymbol{r}$	$\boldsymbol{M}\boldsymbol{v}+\boldsymbol{r}'$	$\{\boldsymbol{M}, \boldsymbol{r}\}$	$\boldsymbol{r}'$
MvM-m	$\{\boldsymbol{M}_i+\boldsymbol{R}_i\}$	$\sum_i \boldsymbol{M}_i\boldsymbol{v}_i+\boldsymbol{r}'$	$\{\boldsymbol{v}_i, \boldsymbol{R}_i\}$	$\boldsymbol{r}'$

3. Cryptographic Primitives

This section presents details about the cryptographic primitives used in our construction.

3.1 (Fully) Homomorphic Encryption

We require an additively homomorphic asymmetric encryption scheme. In this paper, we prefer the Ring-LWE [8] variant of BGV's scheme [2] whose properties enable us to build practical matrix–vector multiplication protocols.

The setup parameters of BGV's scheme consist of three positive integers m, t, and L where t is an exponent of prime values. In our construction, we specify m as an exponent of 2 for efficiency concerns. The message space of this scheme is given as a ring $\mathbb{A}_t := \mathbb{Z}_t[X]/(X^m + 1)$.

We give brief descriptions related to the scheme. Let $(\mathsf{sk}, \mathsf{pk})$ be a key pair generated with parameters m, t, and L. For any element $A, B \in \mathbb{A}_t$, we leverage the following properties of BGV's scheme in our construction.

- Asymmetric scheme:
  $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(A)) = A \mod (X^m + 1, t)$
- Additive homomorphism:

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(A) \oplus \mathsf{Enc}_{\mathsf{pk}}(B)) = A + B \mod (X^m + 1, t)$$
$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(A) \oplus B) = A + B \mod (X^m + 1, t)$$

- Multiplication with scalars:

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(A) \otimes B) = A \times B \mod (X^m + 1, t)$$

The operators $\oplus$ and $\otimes$ respectively indicate homomorphic addition and homomorphic multiplication. Also, we write $\ominus$ to denote the homomorphic subtraction. Indeed, the BGV scheme also supports multiplication of encrypted values:

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(A) \otimes \mathsf{Enc}_{\mathsf{pk}}(B)) = A \times B \mod (X^m + 1, t).$$

However, we do not use this operation in our construction. For the FHE implementation, we use the HElib library [11].

Furthermore, the input of CNNs includes real numbers, whereas FHE handles integers. As described herein, we use the fixed point presentation for real numbers. For a real number x, we convert it to the integer $\lceil x \cdot 2^{\xi} \rfloor$ using a positive integer ξ. In the following sections, we assume that all real values are converted properly into fixed point integers.

3.2 Forward-Backward Packing

The technique of [7], [17] enables efficient private evaluation of inner products. We give a generalization of this technique and show how to extend this technique to efficient matrix-vector multiplication in Section 4. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_t^m$ be vectors of integers. We introduce two functions $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$ that convert a vector of integers to a polynomial:

$$\overrightarrow{\pi}_m(\boldsymbol{u}) = \sum_{i=0}^{m-1} u_i X^i, \quad \overleftarrow{\pi}_m(\boldsymbol{v}) = \sum_{j=0}^{m-1} v_j X^{m-1-j}.$$

Let α be a scalar. We have the following properties.

$$\overrightarrow{\pi}_m(\boldsymbol{u}) + \overrightarrow{\pi}_m(\boldsymbol{v}) = \overrightarrow{\pi}_m(\boldsymbol{u}+\boldsymbol{v}) \quad \alpha \cdot \overrightarrow{\pi}_m(\boldsymbol{u}) = \overrightarrow{\pi}_m(\alpha \cdot \boldsymbol{u})$$
$$\overleftarrow{\pi}_m(\boldsymbol{u}) + \overleftarrow{\pi}_m(\boldsymbol{v}) = \overleftarrow{\pi}_m(\boldsymbol{u}+\boldsymbol{v}) \quad \alpha \cdot \overleftarrow{\pi}_m(\boldsymbol{v}) = \overleftarrow{\pi}_m(\alpha \cdot \boldsymbol{v}).$$

In other words, we can operate the vector addition and the vector-scalar multiplication with $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$. More importantly, the coefficient of X^{m-1} of $\overrightarrow{\pi}_m(\boldsymbol{u}) \times \overleftarrow{\pi}_m(\boldsymbol{v})$ gives the inner product $\boldsymbol{u}^\top \boldsymbol{v}$ as follows.

$$[\overrightarrow{\pi}_m(\boldsymbol{u}) \times \overleftarrow{\pi}_m(\boldsymbol{v})]_{m-1} = \sum_{\substack{i,j \\ :i+(m-1-j)=m-1}} u_i v_j = \boldsymbol{u}^\top \boldsymbol{v}.$$

Let $\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \in \mathbb{Z}_t^{\tilde{m}}$ be vectors such that $\tilde{m} \geq m$. We use $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$ to compute $\tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{v}}$. To do so, we partition the vectors into γ sub-vectors, that is $\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}^{(0)} \| \cdots \| \tilde{\boldsymbol{u}}^{(\gamma-1)}$ and $\tilde{\boldsymbol{v}} = \tilde{\boldsymbol{v}}^{(0)} \| \cdots \| \tilde{\boldsymbol{v}}^{(\gamma-1)}$. Also, we require $|\tilde{\boldsymbol{u}}^{(i)}| = |\tilde{\boldsymbol{v}}^{(i)}|$ and $|\tilde{\boldsymbol{u}}^{(i)}| \leq m$ for $i \in [\gamma]$. We homomorphically compute the inner product $\tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{v}}$ as followings.

$$R \oplus \sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \otimes \mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)})), \tag{1}$$

where R is a degree-m polynomial such that $[R]_i \xleftarrow{\$} \mathbb{Z}_t$ for $i \in [m]$ except that $[R]_{m-1} = 0$. In the context of 2PC, we can assume that Alice pre-processes his input vector $\tilde{\boldsymbol{v}}$ with $\overleftarrow{\pi}$ before the encryption, so that Bob can operate Eq. 1 with his vector $\tilde{\boldsymbol{u}}$ and Alice's ciphertext(s). The correctness and security of Eq. 1 is given by Theorem 1.

Theorem 1. *From the decryption of Eq. 1, except* $\tilde{\mathbf{u}}^\top \tilde{\mathbf{v}}$ mod t *no other information of* $\tilde{\mathbf{u}}$ *and* $\tilde{\mathbf{v}}$ *can be learnt.*

To prove this theorem, we introduce the following lemma.

Lemma 1. *Let d be a positive integer, and let $\boldsymbol{u}$ and $\boldsymbol{v}$ be integer vectors such that $|\boldsymbol{u}| = |\boldsymbol{v}| \leq d$. For a modulo polynomial $X^d + \beta'$ with any nonzero integer β', we have*

$$[\overrightarrow{\pi}_d(\boldsymbol{u}) \times \overleftarrow{\pi}_d(\boldsymbol{v})]_{d-1} = [\overrightarrow{\pi}_d(\boldsymbol{u}) \times \overleftarrow{\pi}_d(\boldsymbol{v}) \mod (X^d + \beta')]_{d-1}.$$

Proofs defer to Appendix A.1.

Eq. 1 requires $\mathcal{O}(\lceil \tilde{m}/m \rceil)$ homomorphic operations. Recall that m is one of the FHE parameters which is usually larger than 2^{12}. This means that we only need a few homomorphic operations to compute the inner product of vectors with thousands of elements. We designate this technique as forward-backward packing (FB-packing).

3.3 CRT-Packing

Aside from the FB-packing, the CRT-packing presented in [13] is another technique commonly used for developing efficient FHE-based protocols. CRT-packing leverages the polynomial Chinese Remainder Theorem (i.e., CRT) to convert a polynomial vector to an element of $\mathbb{A}_t$. We prefer to focus on the properties of this packing. See [13] for more mathematical details.

Table 2 Summary of the two packing methods.

	Input	Specialization
$\overrightarrow{\pi}, \overleftarrow{\pi}$	vectors of integer	inner product, addition of vectors, vector-scalar multiplication
π_{crt}	vector of polynomials	element-wise addition, element-wise multiplication

Given the parameters t and m of BGV's scheme, we can factorize $X^m + 1$ into ℓ distinct and degree-d polynomials $\{F_j\}$ such that $X^m + 1 = \prod_{j=0}^{\ell-1} F_j \mod t$. In literature, factors $\{F_j\}$ are called plaintext slots. ℓ is the number of slots. We write $\pi_{\mathsf{crt}} : (\mathbb{Z}_{t^d})^\ell \to \mathbb{A}_t$ to denote the CRT-packing function[*1], and write π_{crt}^{-1} as the reversing function.

The most important property of this packing is element-wise operations. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be two length-ℓ vectors of *polynomials*, where $(\boldsymbol{p})_j, (\boldsymbol{q})_j \in \mathbb{Z}_t[X]$. The element-wise polynomial addition and multiplication are given as

$$\left(\pi_{\mathsf{crt}}^{-1}(\pi_{\mathsf{crt}}(\boldsymbol{p}) + \pi_{\mathsf{crt}}(\boldsymbol{q}))\right)_j = (\boldsymbol{p})_j + (\boldsymbol{q})_j \mod (F_j, t)$$
$$\left(\pi_{\mathsf{crt}}^{-1}(\pi_{\mathsf{crt}}(\boldsymbol{p}) \times \pi_{\mathsf{crt}}(\boldsymbol{q}))\right)_j = (\boldsymbol{p})_j \times (\boldsymbol{q})_j \mod (F_j, t),$$

for $j \in [\ell]$. Here $+$ and $\times$ indicate the addition and multiplications of polynomials, respectively. Indeed, we can apply this element-wise operation to vectors of integers since we can view integers as degree-0 polynomials. In this case, we achieve ℓ integer additions and integer multiplications.

If we pre-process vectors with π_{crt}, element-wise vector addition and multiplication can be processed in a single operation via the homomorphic operations. For instance, we obtain ℓ additions from one homomorphic addition:

$$\pi_{\mathsf{crt}}^{-1}\left(\mathsf{Dec}_{\mathsf{sk}}\left(\mathsf{Enc}_{\mathsf{pk}}(\pi_{\mathsf{crt}}(\boldsymbol{p})) \oplus \mathsf{Enc}_{\mathsf{pk}}(\pi_{\mathsf{crt}}(\boldsymbol{q}))\right)\right).$$

In other words, CRT-packing reduces the homomorphic computation time by a factor of $1/\ell$, and reduces the number of ciphertexts by $1/\ell$.

We give a summary of the FB-packing and the CRT-packing in Table 2. We use these packings to convert multiple values to a single element of $\mathbb{A}_t$. This reduces the number of ciphertexts which helps to save the communication bandwidth drastically. In the meantime, the two packings also helps to reduce the computation cost of specific operations, e.g., the inner product and the element-wise operations.

Packing Long Vectors. Suppose the length of the vector $\boldsymbol{a}$ is $|\boldsymbol{a}| > m$. To pack $\boldsymbol{a}$ with the FB-packing, we need to partition $\boldsymbol{a}$ into multiple sub-vectors, that is, $\boldsymbol{a} = \boldsymbol{a}^{(0)} \| \boldsymbol{a}^{(1)} \| \cdots$, where $|\boldsymbol{a}^{(i)}| \leq m$. To encode $\boldsymbol{a}$, we use multiple $\overrightarrow{\pi}_m(\cdot)$, that is, $\overrightarrow{\pi}_m(\boldsymbol{a}^{(0)}), \overrightarrow{\pi}_m(\boldsymbol{a}^{(1)}), \ldots$. We simply write $\overrightarrow{\pi}_m(\boldsymbol{a})$ (resp. $\overleftarrow{\pi}_m(\boldsymbol{a})$) to denote these multiple applications of $\overrightarrow{\pi}_m(\cdot)$ (resp. $\overleftarrow{\pi}_m(\cdot)$).

Similarly, for the case of employing π_{crt} on a long vector $\boldsymbol{b}$ such that $|\boldsymbol{b}| > \ell$, we simply write $\pi_{\mathsf{crt}}(\boldsymbol{b})$ to denote $\pi_{\mathsf{crt}}(\boldsymbol{b}^{(0)}), \pi_{\mathsf{crt}}(\boldsymbol{b}^{(1)}), \ldots$, where $\boldsymbol{b} = \boldsymbol{b}^{(0)} \| \boldsymbol{b}^{(1)} \cdots$ and $|\boldsymbol{b}^{(i)}| \leq \ell$.

[*1] Here $\mathbb{Z}_{t^d}$ indicates that the degree of polynomials is d and the coefficients of the polynomial are from $\mathbb{Z}_t$.

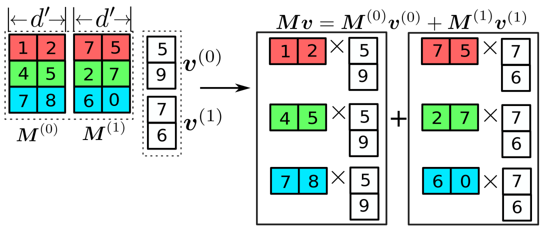

$\leftarrow d' \rightarrow$
$\leftarrow d' \rightarrow$
$M^{(0)}$
$M^{(1)}$
$v^{(0)}$
$v^{(1)}$
$Mv = M^{(0)}v^{(0)} + M^{(1)}v^{(1)}$

Fig. 1 Block-matrix vector multiplication.

4. Proposed Protocols

We now present our constructions of MvM-v and MvM-m.

4.1 MvM-v in Scale

We firstly present a direct approach from the FB-packing. This direct method is practical with respect to computation time but it introduces a large communication overhead. Then we present a novel approach which achieves efficient computation and communication.

4.1.1 A Direct Approach

We can achieve MvM-v by iterations of Eq. 1. The Alice pre-processes its input $\boldsymbol{v}$ with $\overleftarrow{\pi}$. Then Alice sends ciphertext(s) $\mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_m(\boldsymbol{v}))$ to Bob. Let the size of $\boldsymbol{M}$ be $\hat{n}_2 \times \hat{n}_1$. The Bob pre-processes each *row* of $\boldsymbol{M}$ with $\overrightarrow{\pi}$, and thus it obtains $\overrightarrow{\pi}_m(\boldsymbol{M}_{[i:]})$ for $i \in [\hat{n}_2]$. After receiving the ciphertext(s) from Alice, Bob operates Eq. 1 with $\mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_m(\boldsymbol{v}))$ and $\overrightarrow{\pi}_m(\boldsymbol{M}_{[i:]})$ for $i \in [\hat{n}_2]$. Then Bob sends the resulting ciphertexts to Alice. The Alice decrypts all the ciphertexts and obtains $\hat{n}_2$ scalars $\boldsymbol{M}_{[0:]}\boldsymbol{v}, \ldots, \boldsymbol{M}_{[\hat{n}_2-1:]}\boldsymbol{v}$, which forms $\boldsymbol{M}\boldsymbol{v}$.

This direct approach requires $\hat{n}_2\lceil \hat{n}_1/m \rceil$ homomorphic multiplications. We remark that $\lceil \hat{n}_1/m \rceil$ is relatively a small factor (e.g. 2 or 3) because we usually require $m > 2^{12}$ to achieve a reasonable security level (e.g., 80-bit security level) of FHE. The Bob needs to send $\hat{n}_2$ ciphertexts to Alice. This consumes too many bandwidth when $\hat{n}_2$ is large. We remark that this direct approach is suitable for the case that $\hat{n}_1 \gg \hat{n}_2$. We now present a novel approach for the case that both $\hat{n}_1$ and $\hat{n}_2$ are large.

4.1.2 Double-Packing

Notice that Eq. 1 evaluates only a single inner product. This is the reason why the direct approach introduces a large bandwidth cost when tremendous amount of inner products are needed. We propose *double-packing* which combines the FB-packing and the CRT-packing in a delicate way, achieving both good computation and bandwidth efficiency for the large-scale matrix-vector multiplication.

From a high level of view, we compute MvM-v by partitioning the large-scale matrix into several blocks (i.e., submatrices) and thus we decompose one matrix-vector multiplication into several submatrix-subvector multiplications. Then we can apply the FB-packing to each slots of the CRT-packing to evaluate ℓ submatrix-subvector multiplications simultaneously. As a result, we can reduce the number of generated ciphertexts by the factor of $1/\ell$, and thus reduce the bandwidth cost by the factor of $1/\ell$.

Before we presenting the double-packing, we firstly demonstrate that the MvM-v can be viewed as the block-matrix vector multiplication (Figure 1). Let d' be a positive integer, and let $\gamma = \lceil \hat{n}_1/d' \rceil$. The matrix $\boldsymbol{M}$ is vertically partitioned into γ sub-matrices (referred to as blocks) where each block contains at most d' columns of $\boldsymbol{M}$. We write the set of blocks as $\{\boldsymbol{M}^{(b)}\}$. The vector $\boldsymbol{v}$ is also partitioned into γ sub-vectors according to the partition of $\boldsymbol{M}$ so that the length of $\boldsymbol{v}^{(b)}$ is equal to the number of columns of $\boldsymbol{M}^{(b)}$ for $b \in [\gamma]$. Then we can convert the MvM-v to a sum of multiplications of the sub-matrix and the sub-vector, that is, $\boldsymbol{M}\boldsymbol{v} = \sum_{b=0}^{\gamma-1} \boldsymbol{M}^{(b)}\boldsymbol{v}^{(b)}$. Also, $\boldsymbol{M}^{(b)}\boldsymbol{v}^{(b)}$ can be computed through the following inner products

$$\boldsymbol{M}^{(b)}_{[0:]}\boldsymbol{v}^{(b)}, \ldots, \boldsymbol{M}^{(b)}_{[\hat{n}_2-1:]}\boldsymbol{v}^{(b)}. \tag{2}$$

In other words, the computation of $\boldsymbol{M}\boldsymbol{v}$ is decomposed into $\gamma\hat{n}_2$ independent inner products followed by $\gamma - 1$ vector additions.

The first component of our double-packing is to evaluate the $\gamma - 1$ vector additions in the decomposed computation using π_{crt}. Suppose we already have $\{\mathsf{Enc}_{\mathsf{pk}}(\pi_{\mathsf{crt}}(\boldsymbol{z}_b))\}_{b=0}^{\gamma-1}$ where the i-th element of the vector $\boldsymbol{z}_b$ is $(\boldsymbol{z}_b)_i = \boldsymbol{M}^{(b)}_{[i:]}\boldsymbol{v}^{(b)}$ for $i \in [\hat{n}_2]$ (i.e., Eq. 2). Then, the element-wise addition property of π_{crt} allows $\boldsymbol{M}\boldsymbol{v}$ be homomorphically computed through the homomorphic additions $\sum_{b=0}^{\gamma-1} \mathsf{Enc}_{\mathsf{pk}}(\pi_{\mathsf{crt}}(\boldsymbol{z}_b))$. These homomorphic additions result at $\lceil \hat{n}_2/\ell \rceil$ ciphertexts. Thus, we reduce the number of resulting ciphertexts from $\hat{n}_2$ to $\lceil \hat{n}_2/\ell \rceil$, compared with the direct approach.

The second and core component of our double-packing is to compute the inner products of Eq. 2 by employing $\overrightarrow{\pi}$ and $\overleftarrow{\pi}$ inside the plaintext slots of π_{crt}. To demonstrate this employment, we let $\{\boldsymbol{a}_i, \boldsymbol{b}_i\}_{i=0}^{\ell-1}$ be vectors of integers such that $|\boldsymbol{a}_i| = |\boldsymbol{b}_i| = d' < m$. With a proper constraint on d', the following multiplication of polynomials

$$\pi_{\mathsf{crt}}([\overrightarrow{\pi}_{d'}(\boldsymbol{a}_0), \ldots, \overrightarrow{\pi}_{d'}(\boldsymbol{a}_{\ell-1})]) \times \pi_{\mathsf{crt}}([\overleftarrow{\pi}_{d'}(\boldsymbol{b}_0), \ldots, \overleftarrow{\pi}_{d'}(\boldsymbol{b}_{\ell-1})]) \tag{3}$$

correctly give the inner products $\boldsymbol{a}_0^\top\boldsymbol{b}_0, \ldots, \boldsymbol{a}_{\ell-1}^\top\boldsymbol{b}_{\ell-1}$, using Lemma 1 and the element-wise operations of π_{crt}. Thereby, Eq. 2 is evaluated by iterations of Eq. 3.

We now show the constraint of d'. Under some specific combinations of m and t, we can factorize $X^m + 1$ into

$$X^m + 1 = \prod_{j=0}^{\ell-1}(X^d - \beta'_j) \mod t \text{ (all } \beta'_j \neq 0).$$

In other words, the plaintext slots of the CRT-packing become $F_j = X^d - \beta'_j$ for $j \in [\ell]$. According to Lemma 1, we can correctly compute the inner product of length-d vectors using the FB-packing under the modulo polynomial $F_j = X^d - \beta'_j$. That is $[\overrightarrow{\pi}_d(\boldsymbol{a}) \times \overleftarrow{\pi}_d(\boldsymbol{b}) \mod F_j]_{d-1} = \boldsymbol{a}^\top\boldsymbol{b} \mod t$ for any vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_t^d$. Since $m = d\ell$, for the given m and ℓ, the maximum value of d' in Eq. 3 is d.

To achieve this factorization, we need to determine β'_js appropriately. More precisely, we need to make sure that

Table 3 Usable parameters for the double-packing.

m \ ℓ	8	16	32	64	128
2^{12}	113;401	353;673	577;193	641;1153	769;3329
2^{13}	113;338	353;673	577;193	641;1153	7937;3329

Algorithm 1 PrivateMvM-v.

```
Input of Alice: random share of private vector $\tilde{\boldsymbol{v}} = \boldsymbol{v} + \boldsymbol{r}$, private key sk.
Input of Bob: matrix $\boldsymbol{M}$, random share of private vector $\boldsymbol{r}$.
Output of Alice: random share of the output vector $\boldsymbol{M}\boldsymbol{v} + \boldsymbol{r}'$.
Output of Bob: random share of the output vector $\boldsymbol{r}'$.
Notes: $\boldsymbol{v}, \boldsymbol{r} \in \mathbb{Z}_t^{\hat{n}_1}$, $\boldsymbol{M} \in \mathbb{Z}_t^{\hat{n}_2 \times \hat{n}_1}$, $\gamma = \lceil \hat{n}_1/d \rceil$ .
1: The Bob vertically partitions $\boldsymbol{M}$ into $\gamma$ blocks $\{\boldsymbol{M}^{(b)}\}$ where each block contains at most $d$ columns, and Bob then pre-processes the block as $\overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)})$ for $b \in [\gamma]$.
2: The Alice partitions $\tilde{\boldsymbol{v}}$ into $\gamma$ sub-vectors $\{\tilde{\boldsymbol{v}}^{(b)}\}$ and pre-processes $\overleftarrow{\pi}_{\mathsf{w}}(\tilde{\boldsymbol{v}}^{(b)})$ for $b \in [\gamma]$.
3: The Alice computes $\gamma$ ciphertexts $\{\mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_{\mathsf{w}}(\tilde{\boldsymbol{v}}^{(b)}))\}$ and sends them to Bob.
4: The Bob partitions $\boldsymbol{r}$ into $\gamma$ sub-vectors $\{\boldsymbol{r}^{(b)}\}_{b=0}^{\gamma-1}$, and computes $\mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)})) = \mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_{\mathsf{w}}(\tilde{\boldsymbol{v}}^{(b)})) \ominus \overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{r}^{(b)})$.
5: The Bob samples its new share $\boldsymbol{r}' \xleftarrow{\$} \mathbb{Z}_t^{\hat{n}_2}$. Then Bob randomly generates degree-$d$ polynomials $\{R_i\}$ such that $[R_i]_j \xleftarrow{\$} \mathbb{Z}_t$ for $j \in [d]$ except $[R_i]_{d-1} = (\boldsymbol{r}')_i$ for $i \in [\hat{n}_2]$.
6: The Bob homomorphically computes
   $$\pi_{\mathsf{crt}}([R_0, \ldots, R_{\hat{n}_2-1}]) \oplus \sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)}) \otimes \mathsf{Enc}_{\mathsf{pk}}(\overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)})).$$
   Then Bob sends the resulting ciphertexts to Alice.
7: The Alice decrypts all the ciphertexts to obtain $\boldsymbol{M}\boldsymbol{v} + \boldsymbol{r}'$.
```

the multiplicative order of all the β_j's in $\mathbb{Z}_t$ is 2ℓ. We empirically confirmed that finding such β_j's is not difficult for t, m, and ℓ with a reasonable size (see Section 4.3.1).

We now introduce the notation of the double-packing. For the block $\boldsymbol{M}^{(b)}$ and the sub-vector $\boldsymbol{v}^{(b)}$, we write $\overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)})$ and $\overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)})$ to denote the double packing as

$$\overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)}) = \pi_{\mathsf{crt}}\left([\overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[0:]}), \ldots, \overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[\hat{n}_2-1:]})]\right)$$
$$\overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)}) = \pi_{\mathsf{crt}}([\underbrace{\overleftarrow{\pi}_d(\boldsymbol{v}^{(b)}), \ldots, \overleftarrow{\pi}_d(\boldsymbol{v}^{(b)})}_{\hat{n}_2 \text{ copies}}]).$$

The subscript w indicates "double". $\overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)}) \times \overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)})$ (i.e., Eq. 3) evaluates Eq. 2, and thus evaluates $\boldsymbol{M}^{(b)}\boldsymbol{v}^{(b)}$.

4.1.3 Private Evaluation of MvM-v

We now present the private evaluation of MvM-v in Alg. 1 using the double-packing. In Alg. 1, we assume the parameters d and ℓ are shared in advance by Bob and Alice. We also assume that the private input vector of Alice is shared by Alice and Bob, in the from of random shares. That is, Alice holds the share $\tilde{\boldsymbol{v}} = \boldsymbol{v} + \boldsymbol{r}$ while Bob holds the share $\boldsymbol{r}$ where $\boldsymbol{r}$ is a randomly generated vector. At the end of Alg. 1, $\boldsymbol{M}\boldsymbol{v}$ is also possessed in the form of random shares between Alice and Bob. $\{R_i\}$ in Step 5 and R in Eq. 1 are different in two points. One is that the degree of $\{R_i\}$ is d while the degree of R is m. The other is that $[R_i]_{d-1}$ contains Bob's new share $(\boldsymbol{r}')_i$ while $[R]_{m-1} = 0$ in Eq. 1. Thereby, the combination of the element-wise addition of π_{crt} and the vector addition of the FB-packing allows Bob to homomorphically add his new private share $\boldsymbol{r}'$ into the resulting ciphertexts in Step 6.

Theorem 2. *Assume Alice and Bob behave semi-honestly.*

Algorithm 2 PrivateMvM-m.

```
Input of Alice: random shares of private matrices $\{\widetilde{\boldsymbol{M}}_k = \boldsymbol{M}_k + \boldsymbol{R}_k\}_{k=0}^{c-1}$ and private key sk.
Input of Bob: vectors $\{\boldsymbol{v}_k\}_{k=0}^{c}$ and random shares of private matrices $\{\boldsymbol{R}_k\}_{k=0}^{c-1}$.
Output of Alice: random share of the output vector $\sum_{k=0}^{c-1} \boldsymbol{M}_k \boldsymbol{v}_k + \boldsymbol{r}'$.
Output of Bob: random share of the output vector $\boldsymbol{r}'$.
Notes: The size of $\boldsymbol{R}_k, \boldsymbol{M}_k$ is $n' \times h'$. The length of $\boldsymbol{v}_k$ is $h'$.
1: The Alice pre-processes each column of $\widetilde{\boldsymbol{M}}_k$ with $\overrightarrow{\pi}$ and obtains $\overrightarrow{\pi}(\widetilde{\boldsymbol{M}}_{k[:j]})$ for $k \in [c]$ and $j \in [h']$.
2: The Alice computes $\{\mathsf{Enc}_{\mathsf{pk}}\left(\overrightarrow{\pi}(\widetilde{\boldsymbol{M}}_{k[:j]})\right)\}_{k\in[c], j\in[h']}$ and sends the ciphertexts to Bob.
3: For each $\boldsymbol{R}_i$, Bob applies $\overrightarrow{\pi}$ to the columns of $\boldsymbol{R}_i$ to obtain $\overrightarrow{\pi}(\boldsymbol{R}_{i[:j]})$ for $j \in [h']$.
4: The Bob samples its new private share $\boldsymbol{r}' \xleftarrow{\$} \mathbb{Z}_t^{n'}$ and performs the packing to obtain $\overrightarrow{\pi}(\boldsymbol{r}')$.
5: For $k \in [c]$ and $j \in [h']$, Bob computes
   $$\mathsf{Enc}_{\mathsf{pk}}\left(\overrightarrow{\pi}(\boldsymbol{M}_{k[:j]})\right) = \mathsf{Enc}_{\mathsf{pk}}\left(\overrightarrow{\pi}(\widetilde{\boldsymbol{M}}_{k[:j]})\right) \ominus \overrightarrow{\pi}(\boldsymbol{R}_{k[:j]}).$$
6: The Bob homomorphically computes
   $$\overrightarrow{\pi}(\boldsymbol{r}') \oplus \sum_{k=0}^{c-1}\sum_{j=0}^{h'-1} \mathsf{Enc}_{\mathsf{pk}}\left(\overrightarrow{\pi}(\boldsymbol{M}_{k[:j]})\right) \otimes (\boldsymbol{v}_k)_j .$$
   The Bob then sends the resulting ciphertexts to Alice.
7: The Alice obtains $\sum_{i=0}^{c-1} \boldsymbol{M}_i \boldsymbol{v}_i + \boldsymbol{r}'$ after decrypting all the ciphertexts.
```

Then Alg. 1 privately and correctly computes MvM-v. *The Alice learns the share $\boldsymbol{M}\boldsymbol{v} + \boldsymbol{r}'$ but nothing else. The Bob learns the share $\boldsymbol{r}'$ but nothing else.*

The proof defers to Appendix A.2.

Tunable Workloads. We have presented two methods for private evaluation of MvM-v. We show how to decide which approach to use, aiming at less communication overhead.

For the first method, number of the ciphertexts being transferred is $\lceil \hat{n}_1/m \rceil + \hat{n}_2$. For the second method, Alice and Bob, in total, transfer $\lceil \hat{n}_1/d \rceil + \lceil \hat{n}_2/\ell \rceil$ ciphertexts. We can see that when $d \cdot \hat{n}_2 > \hat{n}_1$, it would be better to use the second method. Otherwise, the first method is recommended. Also, we should set ℓ as $\sqrt{\hat{n}_2/\hat{n}_1 \cdot m}$ for the second method to achieve the optimal communication cost by solving $\underset{\ell}{\operatorname{argmin}} \frac{\hat{n}_1}{d} + \frac{\hat{n}_2}{\ell}$ with the constraint $m = d\ell$.

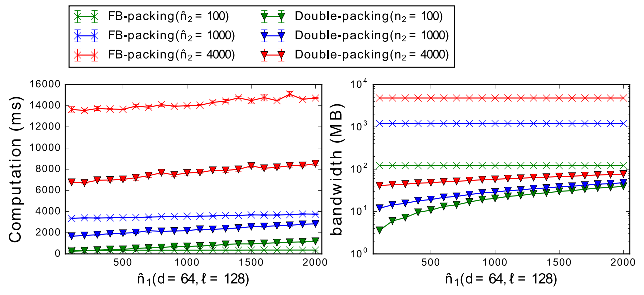
4.2 MvM-M in Scale

In the MvM-m setting, the input of Alice is a set of matrices. We arrange these matrices in a compact form so that we can reduce the number of ciphertexts when we encrypt the matrices. Thus it helps to reduce the communication cost. Consider matrix-vector multiplication $\boldsymbol{U}_k\boldsymbol{v}_k$ where the size of $\boldsymbol{U}_k$ is $n' \times h'$. We simply rewrite it as the equivalent form $\boldsymbol{U}_k\boldsymbol{v}_k = \sum_{j=0}^{h'-1} \boldsymbol{U}_{k[:j]} \cdot (\boldsymbol{v}_k)_j$. We note that $\boldsymbol{U}_{k[:j]} \cdot (\boldsymbol{v}_k)_j$ is a vector-scalar multiplication. Thereby, the required operations to evaluate MvM-m include vector-scalar multiplications and vector additions only. These operations are supported by both of the FB-packing and the CRT-packing. In this work, we prefer to use the FB-packing for MvM-m because the packing time of the FB-packing is faster than that of the CRT-packing.

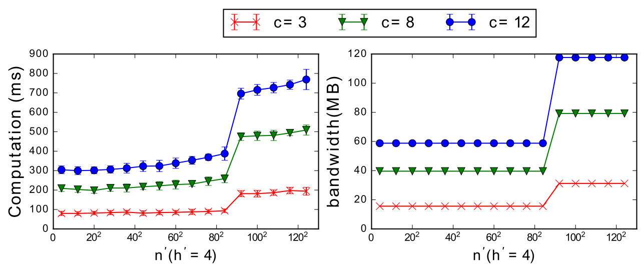
We present MvM-m in Alg. 2. Similar to Alg. 1, we assume the input matrices are already distributed as random

Table 4 Asymptotic analysis of the building blocks.

	computation	communication	
		Alice $\rightarrow$ Bob	Bob $\rightarrow$ Alice
MvM-v (FB-packing)	$\hat{n}_2\lceil\frac{\hat{n}_1}{m}\rceil$	$\lceil\frac{\hat{n}_1}{m}\rceil$	$\hat{n}_2$
MvM-v (double-packing)	$\lceil\frac{\hat{n}_2}{\ell}\rceil\lceil\frac{\hat{n}_1}{d}\rceil$	$\lceil\frac{\hat{n}_1}{d}\rceil$	$\lceil\frac{\hat{n}_2}{\ell}\rceil$
MvM-m	$ch'\lceil\frac{n'}{m}\rceil$	$ch'\lceil\frac{n'}{m}\rceil$	$\lceil\frac{n'}{m}\rceil$



(a) Evaluations of MvM-v using the FB-packing and the double-packing. The matrix size was $\hat{n}_1 \times \hat{n}_2$.



(b) Evaluation of PrivateMvM-m. The matrix size was $n' \times h'$ and the number of matrices was c.

Fig. 2 The computation time of the MvM-v and MvM-m evaluations, and the amount of transferred data. The communication time is excluded.

shares between Alice and Bob. In Step 4, Bob samples its new share $\boldsymbol{r}'$, and homomorphically adds it to the evaluation result in Step 6. We give the following theorem.

Theorem 3. *Suppose Alice and Bob behave semi-honestly. Then Alg. 2 privately and correctly computes MvM-m. The Alice learns the share* $\sum_{i=0}^{c-1} \boldsymbol{M}_i \boldsymbol{v}_i + \boldsymbol{r}'$ *but nothing else. The Bob learns the private share* $\boldsymbol{r}'$ *but nothing else.*

The proof defers to Appendix A.3.

More Compact MvM-m. We can merge the resulting ciphertexts in Step 6 of Alg. 2. To do so, we need to "rotate" the encrypted vector. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be length-d vectors such that $2d \leq m$. It is easily to see that

$$\mathsf{Enc}_{\mathsf{pk}}(\vec{\pi}_m(\boldsymbol{a})) \oplus \left(\mathsf{Enc}_{\mathsf{pk}}(\vec{\pi}_m(\boldsymbol{b})) \otimes X^d\right) = \mathsf{Enc}_{\mathsf{pk}}(\vec{\pi}_m(\boldsymbol{a} \| \boldsymbol{b})).$$

In other words, we can homomorphically merge multiple ciphertexts into one single ciphertext. By doing this, we reduce the number of ciphertexts in Step 6 of Alg. 2 from $c'\lceil n'/m\rceil$ to $\lceil c'n'/m\rceil$. When $c' > n'$, this compact form helps to reduce the bandwidth cost significantly.

4.3 Asymptotic Analysis

We give a brief description of the asymptotic analysis of our building blocks. Let the size of the matrix used in MvM-v be $\hat{n}_2 \times \hat{n}_1$, and n', h' and c be the sizes used in MvM-m. Table 4 summarizes the asymptotic complexity of the building blocks. In this table, we count the number of ciphertexts sent by Alice (i.e., Alice $\rightarrow$ Bob) and the number of ciphertexts sent by Bob (i.e., Bob $\rightarrow$ Alice). Thus, the total bandwidth of the building block can be given as the sum of these two counting. Also, the second column denotes the number of homomorphic operations done by Bob.

4.3.1 Double-packing Revisit

Given a combination of m and t, the double-packing requires at least ℓ distinct values of $\mathbb{Z}_t$ such that the multiplicative order of the value is 2ℓ over $\mathbb{Z}_t$. The number of slots ℓ is determined by the combination of m and t. That is $\ell = m/d$, where $t^d = 1 \mod 2m$.

Given the parameter m, we can perform a grid search for such prime value t that satisfies the requirement. In Table 3, we present some of the usable combinations of m and t for the double-packing. Here, we present two combinations of m and t for each ℓ. For instance, the (m, t) combinations of $(2^{12}, 113)$ and $(2^{12}, 401)$ both satisfy the requirement for the double-packing. These combinations provide $\ell = 8$ plaintext slots, and thus $d = 512$.

5. Experiments

Settings. Our implementations are built with the C++-based FHE library, i.e., HElib [11]. All experiment codes were run by machines with a 2.60GHz Xeon E5-2640 v3 processor and 32GB of RAM. The network speed in our experiments was about 940 Mbps. Multi-threads programming was not employed.

We used the parameters $m = 2^{13}, t = 257^4$, and $L = 5$ of BGV's scheme, which provides at least 128-bit security level according to the security analysis of [4]. The combination of these parameters provides $\ell = 128$ plaintext slots and about 32-bit plaintext space. This combination satisfies the requirement of the double-packing. Under this parameters setting, the size of pk was around 5.4 MB and the size of one FHE ciphertext was about 1.2 MB.

Measurements. We measured the computation time and the bandwidth cost for one call of the FHE-based building blocks. The computation time consists of three: time for encryption on Alice's side, time for homomorphic operations on Bob', and the decryption time on Alice's side. The packing on Bob's side can be considered as a pre-processing, and thus was not included in our experiments.

The bandwidth cost consists of the upstream and downstream cost. The upstream cost means the total amount of data that sent by Alice, and the downstream cost means the total amount of data sent by Bob.

5.1 Protocol Scalability

To see the scalability of our protocols, we separately measured the performance of them with various input sizes.

MvMv. We can instantiate the private MvM-v using either the FB-packing or the double-packing. Fig. 2(a) shows the performances of these two instantiations where d and ℓ denote the parameters used by the double-packing. The matrix size we used was $\hat{n}_2 \times \hat{n}_1$ where $\hat{n}_2 \in \{100, 1000, 4000\}$ and $\hat{n}_1$ was changed from 100 to 2000. In other words, the matrix was changed from a 100×100 small matrix to a huge 4000×2000 matrix.

MvMm. In this experiment, we used $h' = 4$, $c = \{3, 8, 12\}$, and changed n' from 4^2 to 124^2.

5.2 Discussion and Conclusion

From Fig. 2, we can know that when $\hat{n}_2$ is relatively small, the FB-packing can provide a better solution for the private MvM-v. However, the large communication overhead of this method hinders us to use it for the evaluation on a large matrix. On the other hand, the double-packing is extremely efficient even for the case of large matrices. For instance, one private MvM-v using the double-packing on a 4000×2000 matrix only required about 8 seconds and consumed less than 12 MB bandwidth. The same evaluation from the past FHE-based solution [17] might require 29 seconds and consume more than 195 MB bandwidth.

Also, the MvM-m is practical regarding to the computation time. For instance, it cost less than 0.8 seconds to privately compute the MvM-m with 12 matrices each of more than 61504 (i.e., $124^2 \times 4$) elements. Moreover, we can even accelerate the computation time with a multi-threads programming since the convolution is independent on each kernel. We note that the private MvM-m consumes a modest amount of bandwidth which can be reduced by using a more compact FHE implementation, such as [3], [9].

Conclusion. We conclude that our FHE-based approaches are practical enough for large-scale matrix-vector multiplication under the secure two-party computation setting. We consider our approaches are useful for the development of secure protocol of evaluating deep learning algorithm such as convolutional neural networks.

Acknowledgment. This work is supported by JST CREST program "Advanced Core Technologies for Big Data Integration".

References

[1] Mauro Barni, Claudio Orlandi, and Alessandro Piva. A privacy-preserving protocol for neural-network-based computation. In *Proceedings of the 8th workshop on Multimedia and security*, pages 146–151, Geneva, Switzerland, September 26 - 27, 2006.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012*, pages 309–325, Cambridge, MA, USA, January 8-10, 2012.

[3] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - SEAL v2.1. *IACR Cryptology ePrint Archive*, 2017:224, 2017.

[4] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference*, pages 850–867, Santa Barbara, CA, USA, August 19-23, 2012.

[5] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 201–210, NY, USA, June 19-24, 2016.

[6] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge University Press, 2009.

[7] Wen-Jie Lu, Yoshiji Yamada, and Jun Sakuma. Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. *BMC medical informatics and decision making*, 15(5):S1, 2015.

[8] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23, French Riviera, May 30 - June 3, 2010.

[9] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011*, pages 113–124, Chicago, IL, USA, October 21, 2011.

[10] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238, 1999.

[11] Victor Shoup Shai Halevi. HELib. `http://shaih.github.io/HElib`, 2017. Accessed: 2017-04-10.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.

[14] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, pages 139–152, Providence, Rhode Island, USA, June 29 - July 2, 2009.

[15] David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *PoPETs*, 2016(4):335–355, 2016.

[16] Andrew C Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, USA, 3-5 November 1982.

[17] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Secure pattern matching using somewhat homomorphic encryption. In *Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013*, pages 65–76, Berlin, Germany, November 4, 2013.

Appendix

A.1 Proof of Theorem 1

We now give the proof of Theorem 1.

Proof of Correctness. The polynomial

$$(R + \sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \times \overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)})) \mod (X^m + 1, t)$$

is given by the decryption of Eq. 1. According to Lemma 1 we know that

$$[\left(R + \sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \times \overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)})\right) \mod (X^m + 1, t)]_{m-1} =$$
$$[R + \sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \times \overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)}) \mod t]_{m-1} =$$
$$[\sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \times \overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)}) \mod t]_{m-1} = \tilde{\boldsymbol{u}}^\top \tilde{\boldsymbol{v}} \mod t.$$

We use $[R]_{m-1} = 0$ in the last step. □

Proof of Security. The value of $[R]_j$ is distributed uniformly in $\mathbb{Z}_t$ for $0 \leq j < m-1$. Thereby, the value of $[R + \sum_{i=0}^{\gamma-1} \overrightarrow{\pi}_m(\tilde{\boldsymbol{u}}^{(i)}) \times \overleftarrow{\pi}_m(\tilde{\boldsymbol{v}}^{(i)}) \mod (X^m + 1, t)]_j$ is also uniformly distributed in $\mathbb{Z}_t$ for $0 \le j < m-1$. Thereby, no extra information is leaked. $\square$

A.2 Proof of Theorem 2

Proof of Correctness. The element-wise operation property of π_{crt} enables Bob to homomorphically remove its secret share $\boldsymbol{r}$ from Alice's ciphertexts. SP can homomorphically add its new secret share $\boldsymbol{r}'$ to the resulting ciphertexts. It thus suffice to prove that we can learn $\boldsymbol{Mb}$ from $\sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)}) \times \overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)})$.

$$\begin{aligned}
&\sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_{\mathsf{w}}(\boldsymbol{M}^{(b)}) \times \overleftarrow{\pi}_{\mathsf{w}}(\boldsymbol{v}^{(b)}) \\
&= \sum_{b=0}^{\gamma-1} \pi_{\mathsf{crt}}([\overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[0:]}) \times \overleftarrow{\pi}_d(\boldsymbol{v}^{(b)}), \ldots, \\
&\qquad \overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[\hat{n}_2-1:]}) \times \overleftarrow{\pi}_d(\boldsymbol{v}^{(b)})])(\text{element-wise mult.}), \\
&= \pi_{\mathsf{crt}}([\sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[0:]}) \times \overleftarrow{\pi}_d(\boldsymbol{v}^{(b)}), \ldots, \\
&\qquad \sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[\hat{n}_2-1:]}) \times \overleftarrow{\pi}_d(\boldsymbol{v}^{(b)})])(\text{element-wise addition})
\end{aligned}$$

According to the properties of the FB-packing, we know that the coefficient of X^{d-1} of the polynomial $[\sum_{b=0}^{\gamma-1} \overrightarrow{\pi}_d(\boldsymbol{M}^{(b)}_{[j:]}) \times \overleftarrow{\pi}_{d-1}(\boldsymbol{v}^{(b)})]_{d-1}$ is equal to the value $\sum_{b=0}^{\gamma-1} \boldsymbol{M}^{(b)}_{[j:]}\boldsymbol{v}^{(b)}$ for $j \in [\hat{n}_2]$. It is equivalent to $\boldsymbol{Mv}$. $\square$

Proof of Security. First, we prove security against a semi-honest Bob. Intuitively, security against a semi-honest Bob follows from the fact that Bob's view of the execution of Alg. 1 consists only of FHE ciphertexts and independently generated random values. Thus, the security against a semi-honest SP is reduced to the semantic security of the FHE scheme.

The view of Alice in the execution of Alg. 1 consists of $\hat{n}_2$ polynomials of degree d, that is,

$$\mathcal{V}^{\mathsf{MvM\text{-}v}}_{\mathsf{Alice}} = \{R_i + \sum_{b=0}^{\gamma-1} \overrightarrow{\pi}(\boldsymbol{M}^{(b)}_{[0:]}) \times \overleftarrow{\pi}(\boldsymbol{v}^{(b)})\}_{i=0}^{\hat{n}_2-1}.$$

Since the coefficient of the polynomials $\{R_i\}$ was independently sampled from $\mathbb{Z}_t$ (Step 5) uniformly at random, the coefficient of the polynomial $R_i + \sum_{b=0}^{\gamma-1} \overrightarrow{\pi}(\boldsymbol{M}^{(b)}_{[0:]}) \times \overleftarrow{\pi}(\boldsymbol{v}^{(b)})$ is also distributed uniformly on $\mathbb{Z}_t$ for $i \in [\hat{n}_2]$. Thus, Alice's view can be simulated. $\square$

A.3 Proof of Theorem 3

Proof of Correctness. The FB-packing supports vector addition. It enables Bob to homomorphically remove its secret share $\{\boldsymbol{R}_{i[:j]}\}$ from Alice's ciphertexts. Also, it enables Bob to homomorphically add the new secret share $\boldsymbol{r}'$ in the resulting ciphertexts. Thus, it suffice to prove that we can obtain $\sum_{k=0}^{c-1} \boldsymbol{M}_k \boldsymbol{v}_k$ from the computation of

$$\sum_{k=0}^{c-1} \sum_{j=0}^{h'-1} \overrightarrow{\pi}(\boldsymbol{M}_{k[:j]}) \cdot (\boldsymbol{v}_k)_j.$$

The following proof use the vector-scalar multiplication and vector-addition properties of the FB-packing.

$$\begin{aligned}
&\sum_{k=0}^{c-1} \sum_{j=0}^{h'-1} \overrightarrow{\pi}(\boldsymbol{M}_{k[:j]}) \cdot (\boldsymbol{v}_k)_j \\
&= \sum_{k=0}^{c-1} \sum_{j=0}^{h'-1} \overrightarrow{\pi}(\boldsymbol{M}_{k[:j]} \cdot (\boldsymbol{v}_k)_j) \text{ (vector-scalar mult.)} \\
&= \sum_{k=0}^{c-1} \overrightarrow{\pi}(\sum_{j=0}^{h'-1} \boldsymbol{M}_{k[:j]} \cdot (\boldsymbol{v}_k)_j) \text{ (vector-addition)} \\
&= \sum_{k=0}^{c-1} \overrightarrow{\pi}(\boldsymbol{M}_k \boldsymbol{v}_k) \text{ (mat-vec mult. by column vectors)} \\
&= \overrightarrow{\pi}(\sum_{k=0}^{c-1} \boldsymbol{M}_k \boldsymbol{v}_k) \text{ (vector addition).}
\end{aligned}$$

This completes our proof. $\square$

Proof of Security. The proof for the security of Alg. 2 against a semi-honest Bob is similar to that of Alg. 1. Bob's view in the real execution of Alg. 2 consists of two components: ch' ciphertexts sent by Alice, and a vector $\boldsymbol{r}'$ which is sampled from $\mathbb{Z}_t^{n'}$ uniformly at random.

$$\mathcal{V}^{\mathsf{MvM\text{-}m}}_{\mathsf{Bob}} = \{\{\mathsf{Enc}_{\mathsf{pk}}(\overrightarrow{\pi}(\tilde{\boldsymbol{M}}_{k[:j]}))\}_{k \in [c], j \in [h']}, \boldsymbol{r}'\}.$$

By semantic security of the FHE scheme, $\{\mathsf{Enc}_{\mathsf{pk}}(\overrightarrow{\pi}(\tilde{\boldsymbol{M}}_{k[:j]}))\}$ is computationally indistinguishable from independent encryption of 0.

The view of Alice in the execution of Alg. 2 consists only of a length-n' vector $\sum_{k=0}^{c-1} \boldsymbol{M}_k \boldsymbol{v}_k + \boldsymbol{r}'$. Since the vector $\boldsymbol{r}'$ was sampled from $\mathbb{Z}_t^{n'}$ uniformly at random (Step 4), the vector $\sum_{k=0}^{c-1} \boldsymbol{M}_k \boldsymbol{v}_k + \boldsymbol{r}'$ is also distributed uniformly on $\mathbb{Z}_t^{n'}$.

Thus, we can construct simulators that can simulate views that are computationally indistinguishable to the views of Alice and Bob in the real execution of Alg. 2. $\square$