TRANSACTIONS ON DATA PRIVACY 15 (2022) 193–211

Pixel-Batched Homomorphic Encryption for Secure-at-Rest Image Convolution

Chris R. Giannella*, **Adrian V. Mariano***, **James H. Tanis***

*The MITRE Corp., 7515 Colshire Dr. McLean, VA 22102, USA.

E-mail: cgiannella@mitre.org, adrian@mitre.org, jhtanis@mitre.org

Received 2 March 2022; accepted 30 July 2022

Abstract. We consider a secure image processing workflow where the data owner keeps their raw imagery encrypted at all times to reduce the risk of sensitive information exposure in the event of a data breach. However, the owner also makes unencrypted, convolution-filtered versions of the images available to consumers (upon request) who specify a convolution kernel. To acheive this workflow, we utilize homomorphic encryption through the Paillier cryptosystem. We start by considering a simple approach where each pixel is encrypted separately. To reduce overall computation, we next develop a more complex approach where vectors of pixels are batched before encryption (an idea applied by others to different image processing algorithms). Through simulation experiments, we observe the approach involving batching to require one to two orders of magnitude less computation time.

Keywords. Homomorphic Encryption, Image Convolution

1 Introduction

In 1978 Rivest and colleagues noted that public key cryptosystems, such as RSA, did not allow interesting functions to be computed on encrypted data, but doing so would be worthwhile and possible [22]. In the last ten years, homomorphic encryption research has blossomed motivated, in part, by the following scenario.

Data theft is a serious problem facing both private and public organizations. For example: in 2015, the U.S. Office of Personnel Management announced that tens of millions of private personnel records were stolen by hackers [25]. Two years later, Equifax announced that it had been the victim of a data breach that resulted in the potential theft of personal information of 147 million individuals [5]. One strategy to mitigate the damage in cases like these is for the raw data to be kept in encrypted form and only the results of processing on the data is unencrypted. Homomorphic encryption provides a way to address this

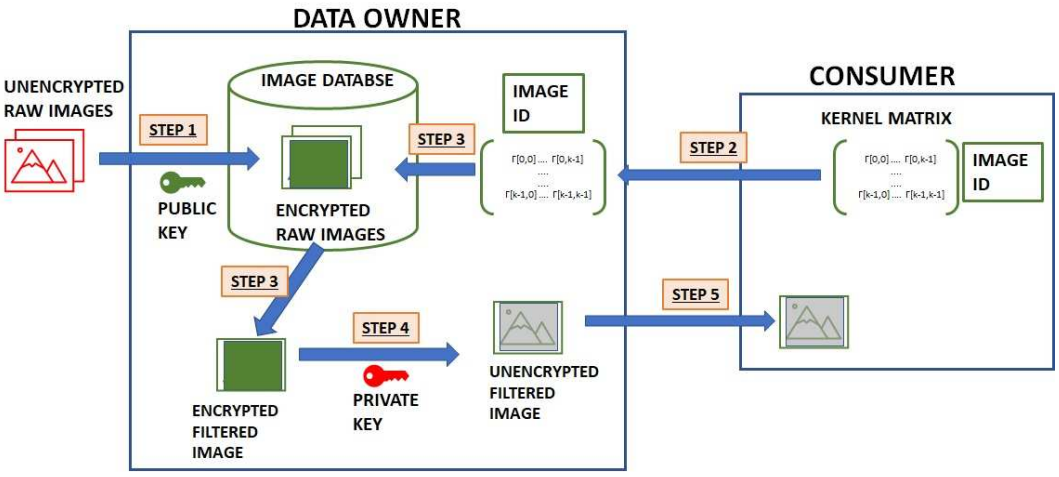

Figure 1: Secure-at-rest image convolution

challenge by allowing computation to be meaningfully performed on encrypted data. This scenario is discussed further in the context of database encryption by Ge and Zdonik [8].

We consider a version of this senario in the context of image procressing. The data owner keeps their raw imagery encrypted at all times (at rest) while providing data consumers, upon request, the unencrypted results of convolution-based filtering of the raw images. It is assumed that the filtered versions of the images are less sensitive than the raw image and therefore can be consumed in unencrypted form. In more detail the workflow, called *secure-at-rest image convolution*, proceeds as follows and is illustrated in Figure 1.

1. The data owner receives raw images and encrypts them using the public key. While the images are at rest in the data onwer's database, they are kept only in encrypted form.
2. A data consumer specifies a convolution kernel and image ID and requests from the owner a filtered version of the raw image with that ID.
3. The owner convolves the encrypted image with the kernel (without decrypting) and produces an encrypted version of filtered image.
4. The data owner decrypts the filtered image using the private key.
5. The data owner delivers the decrypted, filtered image to the consumer.

Provided that the data owner keeps the private key safe from data breach, encrypted images can be exfiltrated from the data owner's database with minimal risk of exposing sensitive information. Questions remain regarding the degree to which the filtered images can be used to reproduce the raw images and what kinds of kernels the data owner should allow. But, to keep this paper tractable, those questions are out of scope.

1.1 Homomorphic Encryption: Overview

In the remainder of this paper, we limit our discussion to workflows with two parties, one of whom owns the data, and one round of communication between them. Other workflows

(beyond the scope of this paper) require more than two parties [15] or multiple rounds of communication between the parties.

Many cryptosystems have been developed allowing operations to be performed homomorphically on encrypted data. A good survey was written by Acar *et al.* [1]. Prior to 2009, homomorphic cryptosystems could be grouped into two categories: *Partially Homomorphic Encryption (PHE)* and *Leveled Fully Homomorphic Encryption (LFHE)*. Partially homomorphic encryption systems allow one operation, addition or multiplication, to be performed homomorphically on pairs of ciphertexts an unlimited number of times (Homomorphic operations between pairs of ciphertexts and plaintexts are also allowed). Details regarding one such system, the Paillier cryptosystem [20], are provided in 3.2. Leveled fully homomorphic encryption systems allow both addition and multiplication between pairs of ciphertext to be performed homomorphically but each application produces noise. The noise accumulates and eventually renders an accurate decryption unlikely. These systems provide a noise budget wherein accurate decryption can be assured provided the accumulation remains within budget.

Gentry achieved a breakthrough in 2009 by constructing the first *Fully Homomorphic Encryption Scheme (FHE)* [9] by modifying an earlier PHE scheme by Goldreich *et al.* [10] to be leveled, fully homomorphic and devised an ingenious technique called bootstrapping to reduce noise to zero. Accurate decryption can be guaranteed with an unlimited number of homomorphic computations provided that bootstrapping is periodically performed. However, bootstrapping is extremely computationally costly rendering its use infeasible. As such, FHE systems are still largely theoretical constructs.

Partially homomorphic encryption systems are less computationally costly than LFHE systems. Moreover, PHE systems are simpler to use as algorithmic building blocks since they do not require a noise budget to be maintained (PHE systems are noise-free). Several papers have been published regarding the application of PHE and LFHE systems to the development of secure image processing algorithms - a summary is provided in Section 2. In this paper, we focus on applying one common PHE system, Paillier, to secure image processing, specifically, image convolution.

2 Related Work: Secure Image Processing

2.1 Leveled Fully Homomorphic Encryption Systems

As discussed earlier, LFHE systems retain the computational power of FHE systems but suffer from noise accumulation. As a result, the implementation of image processing algorithms using LFHE systems are complicated by the need to avoid noise accumulation exceeding a fixed budget. Despite this, researchers have utilized LFHE systems to implement a range of image processing algorithms that can operate on encrypted data. Several efforts have been made at implementing encrypted image classification via convolutional neural networks (CNNs): [2], [4], [11], [14]. A primary challenge faced by these efforts involves non-linear functions, *e.g.* sigmoid, that are used within neural networks. Other efforts have focused on implementing encrypted image feature extraction, *e.g.* SIFT, HOG, Hahn Moments: [12], [29], [31]. Finally, other efforts have focused on implementing a variety of low-level image processing algorithms, *e.g.* image scaling, image JPEG compression/decompression: [7], [13], [30].

2.2 Partially Homomorphic Encryption Schemes

Compared to LFHE systems, PHE systems are simpler to use due to their lack of noise accumulation. The Paillier cryptosystem, a prominent PHE system, has been used to implement several lower-level image processing algorithms on encrypted data: image scaling [17], the discrete cosine transformation [3], discrete wavelet transformations [32], and convolution [33].

In an effort keep the computational power of LFHE systems without the difficulty of noise, some researchers have attempted to develop methods for implementing numeric comparison using a noise-free PHE system. Doing so would extend the class of algorithms that can be implemented using PHE systems. Hsu *et al.* [12] claimed to have an implementation of numeric comparison and SIFT feature extraction on encrypted data utilizing Paillier. However, Schneider [23] argued that Hsu's approach is not secure. Separately, Li *et al.* [16] claimed to have an implementation of comparison which was utilized in [26] for secure medical image analysis and in [28] for secure face matching. However, Wang *et al.* [27] argued that Li's approach is insecure.

2.2.1 Most Relevant

Ziad *et al.* [33] considered convolution (as well as a few other image processing methods) over encrypted data using the Paillier system. However, in their approach, each pixel was encrypted separately and a straight-forward algorithm was developed. Their approach is essentially the same as the non-batched approach discussed in Section 4. Bianchi *et al.* [3] used the Paillier system and developed a pixel-batching approach to improve the efficiency of the Discrete Cosine Transformation (DCT) on encrypted data. Later, Mohanty *et al.* [17] used Paillier and applied pixel-batching to improve the efficiency of bilinear scaling on encrypted data. In both cases, the core pixel-batching idea was to represent integer vectors as base B integers (with B suitably chosen). We utilize the same idea, but, owing to the differences in application, our overall approach differs from theirs. Specifically, expressing convolution on top of batching is different than expressing the DCT or bilinear scaling. Ge and Zdonik used Paillier for computing aggregate queries on an encrypted database [8]. They developed a batching approach for speeding up computation of encrypted column sums. As before, our overall approach is different owing to differences in application.

Batching approaches have also been developed for LFHE systems and they can be used for secure at rest image convolution. We chose Paillier instead because of its relative simplicity. We discuss this choice further in 6.1.1.

3 Preliminaries

Let $\mathbb{Z}_{>0}$ denote the positive integers. Given $N \in \mathbb{Z}_{>0}$, let $\mathbb{Z}_N$ denote the set $\{z \in \mathbb{Z} : 0 \leq z < N\}$ and let $\mathbb{Z}_N^*$ denote the set $\{z \in \mathbb{Z}_{>0} : z < N \text{ and } gcd(z, N) = 1\}$. Given $z \in \mathbb{Z}_N^*$ let z^{-1} denote the multiplicative inverse of z modulo N. $\mathbb{Z}_{N^2}^*$ is defined similarly.

We use Python-style indexing notation to denote sub-matrices. Let M denote an $n \times m$ matrix and consider indices $a_1 < a_2$ and $b_1 < b_2$. We use $M[a_1 : a_2, b_1 : b_2]$ to denote the $(a_2 - a_1) \times (b_2 - b_1)$ sub-matrix from corner entry $M[a_1, b_1]$ to opposite corner entry $M[a_2 - 1, b_2 - 1]$. We use $M[a_1, b_1 : b_2]$ ($M[a_1 : a_2, b_1]$) to denote the one-row (one-column) sub-matrix with entries $M[a_1, b_1]$ to $M[a_1, b_2 - 1]$ ($M[a_1, b_1]$ to $M[a_2 - 1, b_1]$).

3.1 Image Convolution

Convolution is a commonly employed linear filtering operation [21] §13.1.3. Let Υ denote an $n \times m$ image (a matrix) where each pixel assumes a greyscale value in $\{0, 1, 2, \cdots, 255\}$. Let Γ denote a $k \times k$ kernel matrix ($k < n, m$ and k odd) whose entries are rational numbers. Let $\Upsilon * \Gamma$ denote matrix convolution. For simplicity, we define convolution ignoring edge effects, namely, as the $(n - k + 1) \times (m - k + 1)$ matrix whose (i, j) entry is

$$(\Upsilon * \Gamma)[i, j] \stackrel{\text{def}}{=\!=} \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \Upsilon[i + a, j + b]\Gamma[a, b]. \tag{1}$$

We compute convolution using a shift-add algorithm (Algorithm 1) with an optimization to exploit the fact that kernel entries may not be unique. The shift-add algorithm without the optimization would start by multiplying Υ by each kernel entry regardless of the fact that some of the entries may have the same value.[1] We use the dictionary KerMap to avoid this redundancy by recording for each *unique* kernel entry value, the multiplication of Υ by that value.

Data: Υ,Γ
Result: C, the convolution of Υ and Γ
KerMap $\leftarrow \{\}$;
$C \leftarrow (n - k + 1) \times (m - k + 1)$ zero matrix.
for $a = 0 \cdots k - 1$ **do**
  for $b = 0 \cdots k - 1$ **do**
    if $\Gamma[a, b] \notin$ *KerMap.keys()* **then**
      KerMap$[\Gamma[a, b]] \leftarrow \Gamma[a, b]\Upsilon$
    end
  end
end
for $a = 0 \cdots k - 1$ **do**
  for $b = 0 \cdots k - 1$ **do**
    $C \leftarrow C +$ KerMap$[\Gamma[a, b]][a : a + n - k + 1, b : b + m - k + 1]$
  end
end

Algorithm 1: Shift-add convolution with an optimization.

3.2 The Paillier Cryptosystem

The security of the Paillier cryptosystem [20] is based on the conjectured computational difficulty of solving the composite residuosity problem. The level of security is controlled by a non-negative integer N which is the product of two large primes p and q chosen to have certain properties, *e.g.* are close in size. Larger values of N offer greater security at the expense of more computation required for all operations. The plaintext and ciphertext spaces are $\mathbb{Z}_N$ and $\mathbb{Z}^*_{N^2}$, respectively.

Key generation: Let λ denote the least common multiple of $p - 1$ and $q - 1$.

[1] The non-optimized version of shift-add is sketched in [19] §3.2.1.

- The encryption key is (N, g) with g drawn uniformly from

$$\left\{\hat{g} \in \mathbb{Z}^*_{N^2} : \left\lfloor \frac{\hat{g}^\lambda \pmod{N^2} - 1}{N} \right\rfloor \in \mathbb{Z}^*_N \right\}. \tag{2}$$

- The decryption key is (λ, μ) with

$$\mu \stackrel{\text{def}}{=\!=} \left\lfloor \frac{g^\lambda \pmod{N^2} - 1}{N} \right\rfloor^{-1}. \tag{3}$$

Encryption: Given plaintext z, select r uniformly from $\left\{\hat{r} \in \mathbb{Z}^*_{N^2} : \hat{r} < N\right\}$, then compute

$$\text{Enc}\,(z) \stackrel{\text{def}}{=\!=} g^z r^N \pmod{N^2}. \tag{4}$$

Decryption: Given ciphertext $\hat{z}$, compute

$$\text{Dec}\,(\hat{z}) \stackrel{\text{def}}{=\!=} \left\lfloor \frac{\hat{z}^\lambda \pmod{N^2}}{N} \right\rfloor \mu \pmod{N}. \tag{5}$$

Homomorphic Computation: Given ciphertexts $\hat{z}_1, \hat{z}_2$, let $\hat{z}_1 \oplus \hat{z}_2$ denote $\hat{z}_1 \hat{z}_2 \pmod{N^2}$ which is addition performed in ciphertext space. Given plaintext z let $\hat{z}_1 \otimes z$ (or $z \otimes \hat{z}_1$) denote $\hat{z}_1^z \pmod{N^2}$ which is constant multiplication performed in ciphertext space. For any plaintexts z_1, z_2:

$$\text{Dec}\,(\text{Enc}\,(z_1) \oplus \text{Enc}\,(z_2)) = (z_1 + z_2) \pmod{N} \tag{6}$$

and

$$\text{Dec}\,(\text{Enc}\,(z_1) \otimes z) = z_1 z \pmod{N}. \tag{7}$$

Unlike LFHE systems, there is no limit to the number of times $\oplus$ and $\otimes$ can be performed and still decrypt to the correct result (modulo N).

4 Homomorphic Encryption for Secure-at-Rest Image Convolution

Since the plaintext space consists of only non-negative integers, accommodations must be made to handle rational numbers. First, we split the kernel into two parts Γ_+ and Γ_-, both $k \times k$ matrices:

$$\Gamma_+[i,j] \stackrel{\text{def}}{=\!=} \begin{cases} \Gamma[i,j] & \text{if } \Gamma[i,j] > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and

$$\Gamma_-[i,j] \stackrel{\text{def}}{=\!=} \begin{cases} -\Gamma[i,j] & \text{if } \Gamma[i,j] < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Second, we define a non-negative integer matrix $\widehat{\Gamma_+}$ by choosing a scale factor, $\sigma_+ \in \mathbb{Z}_{>0}$, then scaling up each entry in Γ_+ by σ_+ and taking the ceiling. We use a similar process

to define a non-negative integer matrix $\widehat{\Gamma_-}$ through a potentially different scale factor, σ_-. Convolution is performed separately for $\widehat{\Gamma_+}$ and $\widehat{\Gamma_-}$ and each resulting pixel is divided by σ_+ and σ_-, respectively. The resulting two matrices are added to produce an approximation to $\Upsilon * \Gamma$. Larger values of σ_+ and σ_- produce a better approximation at the expense of increased computation due to the larger size of the entries in $\widehat{\Gamma_+}$ and $\widehat{\Gamma_-}$. To strike a balance, we fix an error tolerance $\epsilon > 0$ and choose the smallest values of σ_+ and σ_- to ensure the approximation error is bounded by ϵ.

More precisely, given a predefined error tolerance $\epsilon > 0$, we define $\widehat{\Gamma_+}$ and σ_+ as follows.

$$\sigma_+ \stackrel{\text{def}}{=\!=} \underset{\sigma \in \mathbb{Z}_{>0}}{argmin} \left\{ \max_{0 \le a,b < k} \left\{ \left| \frac{\lceil \sigma \Gamma_+[a,b] \rceil}{\sigma} - \Gamma_+[a,b] \right| \right\} \le \frac{\epsilon}{255k^2} \right\}. \tag{10}$$

Given $a, b \in \{0, \ldots, k-1\}$,

$$\widehat{\Gamma_+}[a,b] \stackrel{\text{def}}{=\!=} \lceil \sigma_+ \Gamma_+[a,b] \rceil \tag{11}$$

We define $\widehat{\Gamma_-}$ and σ_- in similar fashion and approximate $\Upsilon * \Gamma$ by

$$\overline{\Upsilon * \Gamma} \stackrel{\text{def}}{=\!=} \frac{\widehat{\Gamma_+} * \Upsilon}{\sigma_+} - \frac{\widehat{\Gamma_-} * \Upsilon}{\sigma_-}. \tag{12}$$

The following result shows that the approximation error, defined as the maximum absolute difference between matrix entries, is bounded by ϵ.

Theorem 1. Given $0 \le i \le n-k$ and $0 \le j \le m-k$, let $E_{i,j} \stackrel{\text{def}}{=\!=} \left| \left(\overline{\Upsilon * \Gamma}\right)[i,j] - (\Upsilon * \Gamma)[i,j] \right|$. Then, $E_{i,j} \le \epsilon$.

Proof. Let $\mathcal{G}_{\ge}$ and $\mathcal{G}_{<}$ denote the sets of index pairs $\{(a,b) : 0 \le a, b \le k-1 \text{ and } \Gamma[a,b] \ge 0\}$ and $\{(a,b) : 0 \le a, b \le k-1 \text{ and } \Gamma[a,b] < 0\}$, respectively. Plugging in definitions and algebraic manipulations yields that:

$$E_{i,j} = \left| \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \Upsilon[i+a, j+b] \left(\frac{\widehat{\Gamma_+}[a,b]}{\sigma_+} + \frac{\widehat{\Gamma_-}[a,b]}{\sigma_-} - \Gamma[a,b] \right) \right| \tag{13}$$

$$= \left| \sum_{(a,b) \in \mathcal{G}_{\ge}} \Upsilon[i+a, j+b] \left(\frac{\widehat{\Gamma_+}[a,b]}{\sigma_+} - \Gamma[a,b] \right) + \sum_{(a,b) \in \mathcal{G}_{<}} \Upsilon[i+a, j+b] \left(\frac{\widehat{\Gamma_-}[a,b]}{\sigma_-} - \Gamma[a,b] \right) \right|. \tag{14}$$

The triangle inequality and the definitions of σ_+ and σ_- imply

$$(14) \le \sum_{(a,b) \in \mathcal{G}_{\ge}} \Upsilon[i+a, j+b] \left| \frac{\widehat{\Gamma_+}[a,b]}{\sigma_+} - \Gamma[a,b] \right| + \sum_{(a,b) \in \mathcal{G}_{<}} \Upsilon[i+a, j+b] \left| \frac{\widehat{\Gamma_-}[a,b]}{\sigma_-} - \Gamma[a,b] \right| \tag{15}$$

$$\le \sum_{(a,b) \in \mathcal{G}_{\ge}} \Upsilon[i+a, j+b] \frac{\epsilon}{255k^2} + \sum_{(a,b) \in \mathcal{G}_{<}} \Upsilon[i+a, j+b] \frac{\epsilon}{255k^2} \tag{16}$$

$$= \left(\frac{\epsilon}{255k^2} \right) \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \Upsilon[i+a, j+b]. \tag{17}$$

Finally, the fact that all entries of Υ are no greater than 255 implies that $E_{i,j}$ is bounded above by ϵ, as needed.

□

4.1 Data Owner and Consumer's Workflow

The consumer selects ϵ and computes $\widehat{\Gamma_+}$, σ_+, $\widehat{\Gamma_-}$, σ_- as described earlier and sends these matrices to the data owner.

The data owner sets N to a value of their choosing and computes Enc (Υ), the $n \times n$ matrix of ciphertexts. Next, the owner computes M_+ and M_- by applying Algorithm 2 to Enc (Υ), $\widehat{\Gamma_+}$ and Enc (Υ), $\widehat{\Gamma_-}$, respectively. Finally, the data owner computes and sends the following back to the consumer

$$\frac{\text{Dec}\,(M_+)}{\sigma_+} - \frac{\text{Dec}\,(M_-)}{\sigma_-} = \overline{\Upsilon * \Gamma}. \tag{18}$$

Data: Enc (Υ), $\widehat{\Gamma_\pm}$ a $k \times k$ non-negative integer matrix.
Result: $M_\pm$ a $(n-k+1) \times (m-k+1)$ matrix of ciphertexts.
KerMap $\leftarrow \{\}$;
for each (i,j), compute Enc (0) and assign to $M_\pm[i,j]$.
for $a = 0 \cdots k-1$ **do**
  for $b = 0 \cdots k-1$ **do**
    if $\widehat{\Gamma_\pm}[a,b] \notin$ *KerMap.keys()* **then**
      KerMap$[\widehat{\Gamma_\pm}[a,b]] \leftarrow \widehat{\Gamma_\pm}[a,b] \otimes$ Enc (Υ)
    end
  end
end
for $a = 0 \cdots k-1$ **do**
  for $b = 0 \cdots k-1$ **do**
    $M_\pm \leftarrow M_\pm \oplus$ KerMap$[\widehat{\Gamma_\pm}[a,b]][a : a+n-k+1, b : b+m-k+1]$
  end
end

Algorithm 2: Shift-add convolution over encrypted data.

5 Pixel-Batching for Improved Efficiency

The efficiency of Algorithm 2 can be improved by recognizing that element-wise encryption of Υ is wasteful, namely, each pixel is encoded as a much larger ciphertext. Batching the encryption of pixels can overcome this problem. The idea is to represent a vector of pixels $(p_0, \cdots, p_{r-1})$ using a single plaintext. We define a base B (sufficiently large as discussed later) number, Bat$_B(p_0, \cdots, p_{r-1})$, whose digits are $p_0, \cdots, p_{r-1}$.

$$\text{Bat}_B\,(p_0, \cdots, p_{r-1}) \stackrel{\text{def}}{=\!=} \sum_{j=0}^{r-1} p_j B^j. \tag{19}$$

To recover the vector, we perform a series of modulos and divisions by B to extract the digits from Bat$_B\,(p_0, \cdots, p_{r-1})$ and denote this unbatching operation UBat$_B\,(.)$. Next we show how convolution can be performed on top of batching, starting first with an illustration in a simple case.

5.1 Convolution on Batched Vectors, an Illustration: $k = 3$, $n = m = 4$

A key idea is the definition of a length two column vector BConv_+ which depends on $\widehat{\Gamma_+}$ and Υ, but the dependence on Υ is only through its batched rows. We will show how the unbatching operation applied to the i^{th} entry of BConv_+ recovers the i^{th} row of $\widehat{\Gamma_+} * \Upsilon$. Before defining BConv_+, some preliminary definition are in order. We define length four column vector:

$$\mathsf{Bat}_B\left(\Upsilon\right) \stackrel{\text{def}}{=\!=} \begin{bmatrix} \mathsf{Bat}_B\left(\Upsilon[0,0:4]\right) \\ \mathsf{Bat}_B\left(\Upsilon[1,0:4]\right) \\ \mathsf{Bat}_B\left(\Upsilon[2,0:4]\right) \\ \mathsf{Bat}_B\left(\Upsilon[3,0:4]\right) \end{bmatrix}. \tag{20}$$

Given $0 \leq a, b \leq k-1$, we define four-by-three matrix:

$$\mathsf{KM}_B\left(\widehat{\Gamma_+}[a,b]\right) \stackrel{\text{def}}{=\!=} \widehat{\Gamma_+}[a,b] \begin{bmatrix} B^2\mathsf{Bat}_B\left(\Upsilon\right)[0] & B^1\mathsf{Bat}_B\left(\Upsilon\right)[0] & B^0\mathsf{Bat}_B\left(\Upsilon\right)[0] \\ B^2\mathsf{Bat}_B\left(\Upsilon\right)[1] & B^1\mathsf{Bat}_B\left(\Upsilon\right)[1] & B^0\mathsf{Bat}_B\left(\Upsilon\right)[1] \\ B^2\mathsf{Bat}_B\left(\Upsilon\right)[2] & B^1\mathsf{Bat}_B\left(\Upsilon\right)[2] & B^0\mathsf{Bat}_B\left(\Upsilon\right)[2] \\ B^2\mathsf{Bat}_B\left(\Upsilon\right)[3] & B^1\mathsf{Bat}_B\left(\Upsilon\right)[3] & B^0\mathsf{Bat}_B\left(\Upsilon\right)[3] \end{bmatrix}. \tag{21}$$

Finally, we define length two column vector:

$$\mathsf{BConv}_+ \stackrel{\text{def}}{=\!=} \sum_{a=0}^{2} \begin{bmatrix} \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,0]\right)[a,0] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,1]\right)[a,1] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,2]\right)[a,2] \\ \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,0]\right)[a{+}1,0] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,1]\right)[a{+}1,1] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[a,2]\right)[a{+}1,2] \end{bmatrix}. \tag{22}$$

Now we show how the unbatching operation applied to $\mathsf{BConv}_+[0]$ recovers $\left(\widehat{\Gamma_+} * \Upsilon\right)[0,0:2]$. The reasoning for the other entries of BConv_+ is similar. By plugging in definitions, we have that

$$\begin{aligned}
&\mathsf{BConv}_+[0] && (23)\\
&= \mathsf{KM}_B\left(\widehat{\Gamma_+}[0,0]\right)[0,0] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[0,1]\right)[0,1] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[0,2]\right)[0,2] && (24)\\
&+ \mathsf{KM}_B\left(\widehat{\Gamma_+}[1,0]\right)[1,0] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[1,1]\right)[1,1] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[1,2]\right)[1,2] && (25)\\
&+ \mathsf{KM}_B\left(\widehat{\Gamma_+}[2,0]\right)[2,0] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[2,1]\right)[2,1] + \mathsf{KM}_B\left(\widehat{\Gamma_+}[2,2]\right)[2,2] && (26)\\
&= \widehat{\Gamma_+}[0,0]B^2\mathsf{Bat}_B\left(\Upsilon[0,0{:}4]\right) + \widehat{\Gamma_+}[0,1]B^1\mathsf{Bat}_B\left(\Upsilon[0,0{:}4]\right) + \widehat{\Gamma_+}[0,2]B^0\mathsf{Bat}_B\left(\Upsilon[0,0{:}4]\right) && (27)\\
&+ \widehat{\Gamma_+}[1,0]B^2\mathsf{Bat}_B\left(\Upsilon[1,0{:}4]\right) + \widehat{\Gamma_+}[1,1]B^1\mathsf{Bat}_B\left(\Upsilon[1,0{:}4]\right) + \widehat{\Gamma_+}[1,2]B^0\mathsf{Bat}_B\left(\Upsilon[1,0{:}4]\right) && (28)\\
&+ \widehat{\Gamma_+}[2,0]B^2\mathsf{Bat}_B\left(\Upsilon[2,0{:}4]\right) + \widehat{\Gamma_+}[2,1]B^1\mathsf{Bat}_B\left(\Upsilon[2,0{:}4]\right) + \widehat{\Gamma_+}[2,2]B^0\mathsf{Bat}_B\left(\Upsilon[2,0{:}4]\right) && (29)\\
&= \widehat{\Gamma_+}[0,0]\sum_{j=0}^{3}\Upsilon[0,j]B^{j+2} + \widehat{\Gamma_+}[0,1]\sum_{j=0}^{3}\Upsilon[0,j]B^{j+1} + \widehat{\Gamma_+}[0,2]\sum_{j=0}^{3}\Upsilon[0,j]B^{j} && (30)\\
&+ \widehat{\Gamma_+}[1,0]\sum_{j=0}^{3}\Upsilon[1,j]B^{j+2} + \widehat{\Gamma_+}[1,1]\sum_{j=0}^{3}\Upsilon[1,j]B^{j+1} + \widehat{\Gamma_+}[1,2]\sum_{j=0}^{3}\Upsilon[1,j]B^{j} && (31)\\
&+ \widehat{\Gamma_+}[2,0]\sum_{j=0}^{3}\Upsilon[2,j]B^{j+2} + \widehat{\Gamma_+}[2,1]\sum_{j=0}^{3}\Upsilon[2,j]B^{j+1} + \widehat{\Gamma_+}[2,2]\sum_{j=0}^{3}\Upsilon[1,j]B^{j}. && (32)
\end{aligned}$$

Collecting the coefficients on the $B's$ yields

$$\begin{aligned}
\text{BConv}_+[0] &= (\cdots)\,B^0 + (\cdots)\,B^1 && (33)\\
&+ \begin{pmatrix} \widehat{\Gamma_+}[0,0]\Upsilon[0,0] + \widehat{\Gamma_+}[0,1]\Upsilon[0,1] + \widehat{\Gamma_+}[0,2]\Upsilon[0,2] \\ +\widehat{\Gamma_+}[1,0]\Upsilon[1,0] + \widehat{\Gamma_+}[1,1]\Upsilon[1,1] + \widehat{\Gamma_+}[1,2]\Upsilon[1,2] \\ +\widehat{\Gamma_+}[2,0]\Upsilon[2,0] + \widehat{\Gamma_+}[2,1]\Upsilon[2,1] + \widehat{\Gamma_+}[2,2]\Upsilon[2,2] \end{pmatrix} B^2 && (34)\\
&+ \begin{pmatrix} \widehat{\Gamma_+}[0,0]\Upsilon[0,1] + \widehat{\Gamma_+}[0,1]\Upsilon[0,2] + \widehat{\Gamma_+}[0,2]\Upsilon[0,3] \\ +\widehat{\Gamma_+}[1,0]\Upsilon[1,1] + \widehat{\Gamma_+}[1,1]\Upsilon[1,2] + \widehat{\Gamma_+}[1,2]\Upsilon[1,3] \\ +\widehat{\Gamma_+}[2,0]\Upsilon[2,1] + \widehat{\Gamma_+}[2,1]\Upsilon[2,2] + \widehat{\Gamma_+}[2,2]\Upsilon[2,3] \end{pmatrix} B^3 && (35)\\
&+ (\cdots)\,B^4 + (\cdots)\,B^5 && (36)\\
&= (\cdots)\,B^0 + (\cdots)\,B^1 && (37)\\
&+ \left((\widehat{\Gamma_+} * \Upsilon)[0,0]\right) B^2 && (38)\\
&+ \left((\widehat{\Gamma_+} * \Upsilon)[0,1]\right) B^3 && (39)\\
&+ (\cdots)\,B^4 + (\cdots)\,B^5. && (40)
\end{aligned}$$

Therefore, provided that B is large enough, $(\widehat{\Gamma_+} * \Upsilon)[0, 0:2]$ can be extracted from the coefficients of B^2 and B^3 in $\text{BConv}_+[0]$, i.e. $(\widehat{\Gamma_+} * \Upsilon)[0, 0{:}2] = (\text{UBat}_B\,(\text{BConv}_+[0]))[2{:}4]$.

Next, we lift the restrictions that $k = 3$ and $n = m = 4$, expand the reasoning to apply in the general case, and make precise the requirements on the size of B.

5.2 Convolution on Batched Vectors, Full Details in the General Case

Let $\text{Bat}_B\,(\Upsilon)$ be the length n column vector with i^{th} entry $\text{Bat}_B\,(\Upsilon[i, 0 : m])$.

Given $0 \leq a, b \leq k-1$, let $\text{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)$ denote the $n \times k$ matrix whose c^{th} column has i^{th} entry $\widehat{\Gamma_+}[a,b]B^{k-1-c}\text{Bat}_B\,(\Upsilon)\,[i]$.

Finally, define length $n-k+1$ column vector

$$\text{BConv}_+ \stackrel{\text{def}}{=\!=} \sum_{a=0}^{k-1} \begin{bmatrix} \sum_{b=0}^{k-1} \text{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)[a,b] \\ \vdots \\ \sum_{b=0}^{k-1} \text{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)[a+n-k,b] \end{bmatrix} \tag{41}$$

where the outer summation is performed element-wise.

The following result shows, provided that B is large enough: how $\widehat{\Gamma_+} * \Upsilon$ can be recovered from BConv_+, and that each entry of BConv_+ is bounded by $B^{m+k-1} - 1$.

Theorem 2. If $B \geq (255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b]) + 1$, then for any $0 \leq i \leq n-k$:

$$(\widehat{\Gamma_+} * \Upsilon)[i, 0 : (m-k)] = (\text{UBat}_B\,(\text{BConv}_+[i]))[(k-1) : m] \tag{42}$$

and

$$\text{BConv}_+[i] \leq B^{m+k-1} - 1. \tag{43}$$

Proof. Let $\mathcal{P}$ denote the set of index pairs $\{(b,j) : 0 \leq b \leq k-1 \text{ and } 0 \leq j \leq m-1\}$. Plugging in definitions and reordering sums yields:

$$\begin{aligned}
\text{BConv}_+[i] &= \sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b]B^{k-1-b}\text{Bat}_B\left(\Upsilon[a+i,0:m]\right) && (44)\\
&= \sum_{a=0}^{k-1}\sum_{b=0}^{k-1}\sum_{j=0}^{m-1} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,j]B^{k-1+j-b} && (45)\\
&= \sum_{a=0}^{k-1}\sum_{(b,j)\in\mathcal{P}} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,j]B^{k-1+j-b}. && (46)
\end{aligned}$$

Given an integer ℓ, let $\mathcal{P}(\ell)$ denote the subset $\{(b,j) : 0 \leq b \leq k-1 \text{ and } 0 \leq j \leq m-1 \text{ and } j-b=\ell\}$. The following facts are easy to show.

$$\mathcal{P} = \bigcup_{\ell=-(k-1)}^{m-1} \mathcal{P}(\ell). \tag{47}$$

$$\mathcal{P}(\ell_1) \cap \mathcal{P}(\ell_2) = \emptyset \text{ for all } -(k-1) \leq \ell_1 \neq \ell_2 \leq m-1. \tag{48}$$

$$\text{If } -(k-1) \leq \ell \leq -1, \text{ then } \mathcal{P}(\ell) = \{(b, b+\ell) : -\ell \leq b \leq k-1\}. \tag{49}$$

$$\text{If } 0 \leq \ell \leq m-k, \text{ then } \mathcal{P}(\ell) = \{(b, b+\ell) : 0 \leq b \leq k-1\}. \tag{50}$$

$$\text{If } m-k+1 \leq \ell \leq m-1, \text{ then } \mathcal{P}(\ell) = \{(b, b+\ell) : 0 \leq b \leq m-1-\ell\}. \tag{51}$$

The derivation continues:

$$\begin{aligned}
(46) &= \sum_{a=0}^{k-1}\sum_{\ell=-(k-1)}^{m-1}\sum_{(b,j)\in\mathcal{P}(\ell)} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,j]B^{k-1+j-b} && (52)\\
&= \sum_{\ell=-(k-1)}^{m-1}\sum_{a=0}^{k-1}\sum_{(b,j)\in\mathcal{P}(\ell)} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,j]B^{k-1+j-b} && (53)\\
&= \sum_{\ell=-(k-1)}^{-1}\sum_{a=0}^{k-1}\sum_{b=-\ell}^{k-1} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,b+\ell]B^{k-1+\ell} && (54)\\
&+ \sum_{\ell=0}^{m-k}\sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,b+\ell]B^{k-1+\ell} && (55)\\
&+ \sum_{\ell=m-k+1}^{m-1}\sum_{a=0}^{k-1}\sum_{b=0}^{n-1-\ell} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,b+\ell]B^{k-1+\ell}. && (56)
\end{aligned}$$

Facts (47) and (48) imply (52). Splitting the outer sum in (53) and facts (49), (50), and (51) imply (54), (55), and (56). By definition of convolution, the summation (55) can be rewritten to produce:

$$\text{BConv}_+[i] = \sum_{\ell=-(k-1)}^{-1} \sum_{a=0}^{k-1} \sum_{b=-\ell}^{k-1} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,b+\ell]B^{k-1+\ell} \quad (57)$$

$$+ \sum_{\ell=0}^{m-k} (\widehat{\Gamma_+} * \Upsilon)[i,\ell]B^{k-1+\ell} \quad (58)$$

$$+ \sum_{\ell=m-k+1}^{m-1} \sum_{a=0}^{k-1} \sum_{b=0}^{m-1-\ell} \widehat{\Gamma_+}[a,b]\Upsilon[a+i,b+\ell]B^{k-1+\ell}. \quad (59)$$

By assumption, B is larger than any of the coefficients on B^0, B^1, ..., B^{m+k-2} in (57), (58), and (59). Hence $(\widehat{\Gamma_+} * \Upsilon)[i, 0 : (m-k)]$ can be recovered from $\text{BConv}_+[i]$ by extracting the coefficients on B^{k-1}, B^k, ..., B^{m-1}, namely, (42) holds. Furthermore, (43) holds since:

$$\text{BConv}_+[i] \leq (B-1) \sum_{\ell=-(k-1)}^{-1} B^{k-1+\ell} \quad (60)$$

$$+ (B-1) \sum_{\ell=0}^{m-k} B^{k-1+\ell} \quad (61)$$

$$+ (B-1) \sum_{\ell=m-k+1}^{m-1} B^{k-1+\ell} \quad (62)$$

$$= (B-1) \sum_{c=0}^{m+k-2} B^c \quad (63)$$

$$= B^{m+k-1} - 1. \quad (64)$$

□

5.3 Convolution on Encrypted, Batched Vectors

Motivated by earlier definitions, let $\text{Enc}(\text{Bat}_B(\Upsilon))$ denote the length n vector whose i^{th} entry is $\text{Enc}(\text{Bat}_B(\Upsilon)[i])$.

Given $0 \leq a, b \leq k-1$, let $\text{EKM}_B\left(\widehat{\Gamma_+}[a,b]\right)$ denote the $n \times k$ matrix whose c^{th} column has i^{th} entry $\left(\widehat{\Gamma_+}[a,b]B^{k-1-c}\right) \otimes \text{Enc}(\text{Bat}_B(\Upsilon)[i])$.

Finally, define length $n-k+1$ column vector

$$\text{EBConv}_+ \stackrel{\text{def}}{=} \bigoplus_{a=0}^{k-1} \begin{bmatrix} \oplus_{b=0}^{k-1} \text{EKM}_B\left(\widehat{\Gamma_+}[a,b]\right)[a,b] \\ \vdots \\ \oplus_{b=0}^{k-1} \text{EKM}_B\left(\widehat{\Gamma_+}[a,b]\right)[a+n-k,b] \end{bmatrix}. \quad (65)$$

where the outer $\oplus$ operation is performed element-wise. Building on Theorem 2, the final result shows, provided that some constraints on B, N, m, and k are satisfied: how $\widehat{\Gamma_+} * \Upsilon$ can be recovered from EBConv_+.

Theorem 3. If $B \geq (255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b]) + 1$ and $N \geq B^{m+k-1}$, then for any $0 \leq i \leq n-k$:

$$(\widehat{\Gamma_+} * \Upsilon)[i, 0:(m-k)] = (\mathsf{UBat}_B\left(\mathsf{Dec}\left(\mathsf{EBConv}_+[i]\right)\right))[(k-1):m]. \tag{66}$$

Proof. Plugging in definitions, utilizing the homomorphic properties of $\oplus$ and $\otimes$, and applying (44), we have:

$$\begin{aligned}
\mathsf{EBConv}_+[i] &= \oplus_{a=0}^{k-1} \oplus_{b=0}^{k-1} \left(\widehat{\Gamma_+}[a,b] B^{k-1-b}\right) \otimes \mathsf{Enc}\left(\mathsf{Bat}_B\left(\Upsilon\right)[a+i]\right) && (67)\\
&= \oplus_{a=0}^{k-1} \oplus_{b=0}^{k-1} \mathsf{Enc}\left(\widehat{\Gamma_+}[a,b] B^{k-1-b} \mathsf{Bat}_B\left(\Upsilon\right)[a+i] \pmod N\right) && (68)\\
&= \mathsf{Enc}\left(\left(\sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b] B^{k-1-b} \mathsf{Bat}_B\left(\Upsilon\right)[a+i]\right) \pmod N\right) && (69)\\
&= \mathsf{Enc}\left(\left(\sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b] B^{k-1-b} \mathsf{Bat}_B\left(\Upsilon[a+i,:]\right)\right) \pmod N\right) && (70)\\
&= \mathsf{Enc}\left(\mathsf{BConv}_+[i] \pmod N\right). && (71)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathsf{Dec}\left(\mathsf{EBConv}_+[i]\right) &= \mathsf{BConv}_+[i] \pmod N && (72)\\
&= \mathsf{BConv}_+[i] && (73)
\end{aligned}$$

where (73) follows from (43) and the assumption $N \geq B^{m+k-1}$. From Theorem 2, the assumption that $B \geq (255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b]) + 1$, and (73), the desired result follows. □

5.4 Owner and Consumer's Workflow

The consumer selects ϵ and computes $\widehat{\Gamma_+}$, σ_+, $\widehat{\Gamma_-}$, σ_- as described earlier and sends these matrices to the data owner.

The owner sets N to a value of their choosing and sets

$$B \stackrel{\text{def}}{=\!=} \max\left\{255 \sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b], 255 \sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \widehat{\Gamma_-}[a,b]\right\} + 1. \tag{74}$$

The number of image columns, m, must satisfy $N \geq B^{m+k-1}$. Since B, N, and k are fixed, the data owner must ensure that the number of columns is no larger than

$$\hat{m} \stackrel{\text{def}}{=\!=} \left\lfloor \frac{\log N}{\log B} - k + 1 \right\rfloor \tag{75}$$

or else must split the columns of Υ into small enough strips. We will assume that $m > \hat{m}$ since no splitting is required in the case $m \leq \hat{m}$. The data owner creates separate image strips ${}_0\Upsilon, \cdots, {}_r\Upsilon$ where ${}_0\Upsilon$ contains the first $\hat{m}$ columns; ${}_1\Upsilon$ contains columns $\hat{m} - \frac{k-1}{2}$, $\hat{m} - \frac{k-1}{2} + 1, \cdots, 2\hat{m} - \frac{k-1}{2}$; *etc.*. Consecutive entries in ${}_0\Upsilon, \cdots, {}_r\Upsilon$ overlap to accommodate edges being ignored in convolution. Note that each strip, ${}_\ell\Upsilon$, has dimensions $n \times \hat{m}$.

For each $0 \leq \ell \leq r$: the data owner computes $\mathsf{Enc}\left(\mathsf{Bat}_B\left({}_\ell\Upsilon\right)\right)$, a length $\hat{m}$ vector of ciphertexts. The owner computes ${}_\ell\mathsf{EBC}_+$ and ${}_\ell\mathsf{EBC}_-$ by applying Algorithm 3 to $\mathsf{Enc}\left(\mathsf{Bat}_B\left({}_\ell\Upsilon\right)\right)$,

$\widehat{\Gamma_+}$ and $\text{Enc}\left(\text{Bat}_B\left({}_\ell\Upsilon\right)\right)$, $\widehat{\Gamma_-}$, respectively. The owner computes $(n-k+1)\times(\hat{m}-k+1)$ matrix, ${}_\ell\Lambda$, whose i^{th} row is

$$\frac{\text{UBat}_B\left(\text{Dec}\left({}_\ell\text{EBC}_+[i]\right)\right)[k-1:\hat{m}]}{\sigma_+} - \frac{\text{UBat}_B\left(\text{Dec}\left({}_\ell\text{EBC}_-[i]\right)\right)[k-1:\hat{m}]}{\sigma_-}.$$

Below we will show that ${}_\ell\Lambda$ equals $\overline{{}_\ell\Upsilon * \Gamma}$.

Finally, the data owner concatenates ${}_\ell\Lambda$ over all ℓ to produce $\overline{\Upsilon * \Gamma}$ and sends this final result to the consumer.

Data: $\text{Enc}\left(\text{Bat}_B\left({}_\ell\Upsilon\right)\right), \widehat{\Gamma_\pm}$
Result: ${}_\ell\text{EBC}_\pm$ a length $n-k+1$ vector of ciphertexts.
KerMap $\leftarrow \{\}$.
for $i = 0\cdots n-k$, ${}_\ell\text{EBC}_\pm[i] \leftarrow \text{Enc}(0)$.
for $a = 0\cdots k-1$ **do**
  for $b = 0\cdots k-1$ **do**
    if $\widehat{\Gamma_\pm}[a,b] \notin$ *KerMap.keys()* **then**
      $\text{KerMap}[\widehat{\Gamma_\pm}[a,b]] \leftarrow {}_\ell\text{EKM}_B\left(\widehat{\Gamma_\pm}[a,b]\right)$
    end
  end
end
for $a = 0\cdots k-1$ **do**

$$ {}_\ell\text{EBC}_\pm \leftarrow {}_\ell\text{EBC}_\pm \oplus \begin{bmatrix} \bigoplus_{b=0}^{k-1} \text{KerMap}\left[\widehat{\Gamma_\pm}[a,b]\right][a,b] \\ \vdots \\ \bigoplus_{b=0}^{k-1} \text{KerMap}\left[\widehat{\Gamma_\pm}[a,b]\right][a+n-k,b] \end{bmatrix} $$

end

Algorithm 3: Shift-add convolution in batched space.

5.4.1 Correctness Proof

It can be seen that ${}_\ell\text{EBC}_+$ equals ${}_\ell\text{EBConv}_+$ and ${}_\ell\text{EBC}_-$ equals ${}_\ell\text{EBConv}_-$. By definition,

$$B \geq (255\sum_{a=0}^{k-1}\sum_{b=0}^{k-1}\widehat{\Gamma_+}[a,b]) + 1 \tag{76}$$

and $N \geq B^{\hat{m}+k-1}$. Therefore, (66) implies that ${}_\ell\Lambda$ equals

$$\frac{{}_\ell\Upsilon * \widehat{\Gamma_+}}{\sigma_+} - \frac{{}_\ell\Upsilon * \widehat{\Gamma_-}}{\sigma_-} = \overline{{}_\ell\Upsilon * \Gamma} \tag{77}$$

as needed.

6 Empirical Evaluation

The Paillier Homomorphic Encryption Library (PHE) was originally released as an open source project by the Python Software Foundation in December 2014. We used version 1.4.0, released in April 2018 [18]. This library provides a clean interface to carry out encrypted computations based on the Paillier cryptosystem. We conducted experiments comparing the run-times of plaintext convolution with those of Paillier convolution (batched and not). In all experiments, we used a 512×512 greyscale image and six kernels: box blur kernels (all entries of the kernel are the same and sum to one) of sizes 3, 5, and 7 and approximate Gaussian blur kernels of the same sizes - see Table 1. We used only one image since the run times do not depend upon the contents of the image. The image used in all experiments is displayed in Figure 2 (left). In addition, the figure displays the images produced by applying box blur convolutions.

$$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} \quad \frac{1}{256}\begin{bmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{bmatrix} \quad \frac{1}{27777}\begin{bmatrix}1&10&40&64&40&10&1\\10&102&407&645&407&102&10\\40&407&1625&2574&1625&407&40\\64&645&2574&4077&2574&645&64\\40&407&1625&2574&1625&407&40\\10&102&407&645&407&102&10\\1&10&40&64&40&10&1\end{bmatrix}$$

Table 1: Approximate Gaussian blur kernels of sizes 3, 5, and 7.

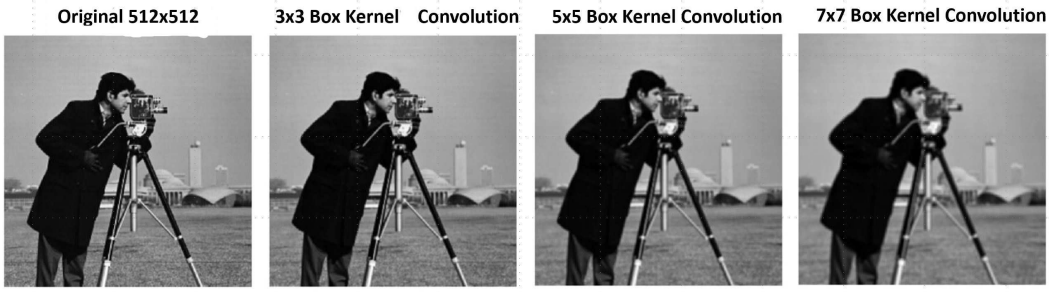

Figure 2: Image used in all experiments (left, ©Massachusetts Institute of Technology, available under `creativecommons.org/licenses/by-nc-sa/3.0`) and those produced by box blur.

For each experiment, we carried out six trials during each of which we computed[2] the run-times of plaintext, non-batched Paillier, and batched Paillier convolution for a fixed kernel, fixed number of bits for N. For experiments with kernels of size three, ϵ was fixed at 0.023; with kernels of size five and seven, ϵ was fixed at 0.125 and 0.637, respectively. We computed the average run-time for each convolution method and its 0.95 confidence intervals under the assumption that run-times were *i.i.d.* Gaussian. We varied the experiments across all combinations of the six kernels and number of bits for N in $\{512, 1024, 2048\}$.

[2]On a Dell Precision 7530 laptop with 16GB of RAM, a 2.6GHz processor running Windows 10 Enterprise (64 bits).

6.1 Results

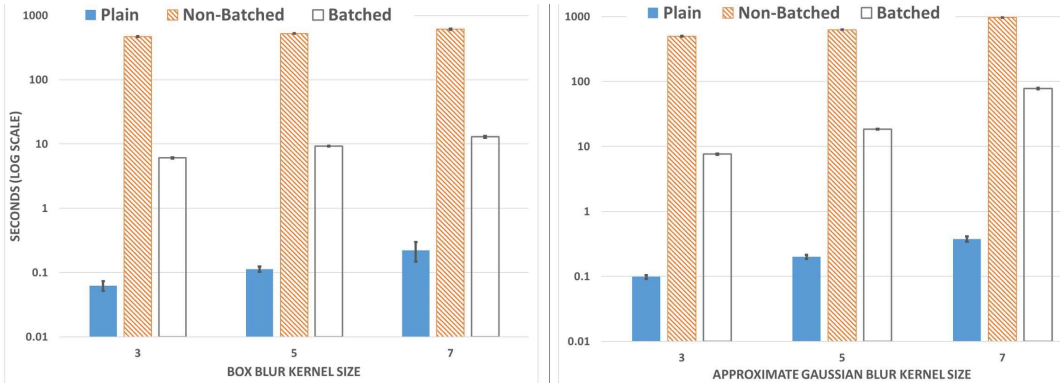

Figure 3: Experiments with varying kernel sizes and the bit length of N fixed at 1024.

Figure 3 shows the results of varying the kernel sizes with the number of bits in N fixed. Batching achieved between one and two orders of magnitude of run-time savings yet still required nearly two orders of magnitude of run-time beyond plaintext convolution. The difference between batching and non-batching run-times was less pronounced for the approximate Gaussian blur kernel than the box blur kernel. This is because the latter requires a smaller scaling factor σ_+.

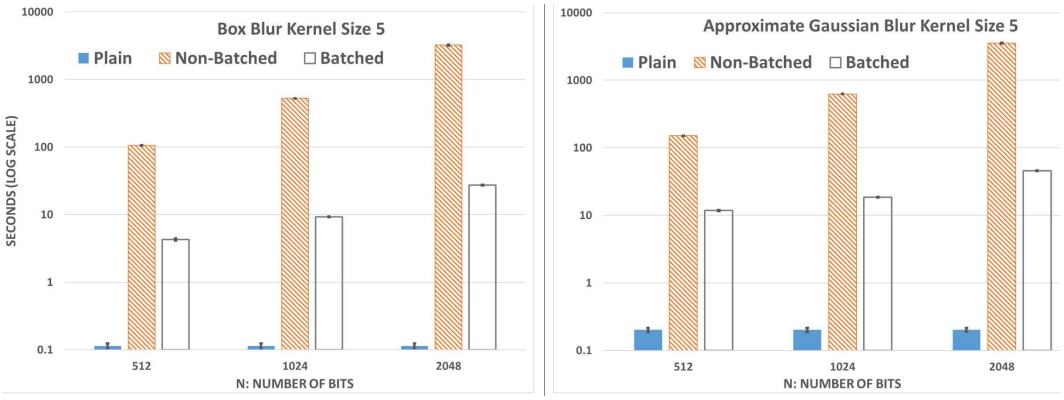

Figure 4: Experiments with varying bit length of N and kernel sizes fixed at five.

Figure 4 shows the results of varying the number of bits in N with the kernel sizes fixed. As expected, the run-time gap between batched and non-batched widened with increasing number of bits in N. This is because the extra bits allowed for greater sized vectors to be batched. Less expected, the run-times for batched convolution increased overall with increasing number of bits. While increasing number of bits allowed larger batching sizes, it also required greater encryption, decryption, and homomorphic computation times. Apparently, the latter effect was more pronounced.

6.1.1 Pixel Batching with LFHE

The Microsoft Simple Encrypted Arithmetic Library (SEAL) [24] provides an implementation of the BFV homomorphic cryptosystem [6] which is based on the Ring Learning with Errors problem. SEAL provides encrypted batch vector encoding and element-wise sums and products (using an approach unrelated to batching as we describe earlier) which can be used to compute image convolution. Because BFV is more complex than Paillier, secure convolution using SEAL is significantly less efficient than our approach using Paillier [24]. However, BFV is more powerful than Paillier (in terms of homomorphic computations supported) and, therefore, supports a broader range of secure image processing.

7 Summary and Future Work

We addressed the problem of secure-at-rest image convolution. We developed a straightforward approach utilizing the Paillier cryptosystem where each pixel is encrypted separately. We improved upon this approach by batching vectors of pixels before encryption. This idea has been applied by others to different image processing algorithms: DCT [3] and bilinear scaling [17]. Our experiments showed that batched convolution required between one and two orders of magnitude less run-time than non-batched convolution. Our experiments also addressed a computation trade-off related to increasing the number of bits in N. On the one hand, increasing the number of bits allows batching to be more effective since the length of the batched vectors is increased. On the other hand, increasing the number of bits requires greater encryption, decryption, and homomorphic computation times. Our experiments showed the latter effect to be more pronounced as the run times of batched convolution grew with increasing number of bits in N.

Zheng *et al.* [32] developed an interesting idea allowing, in some circumstances, factors to be removed from Paillier encrypted values. In our case, if we assume that $\sigma_+\Gamma_+[a,b]$ is an integer for all a and b and we assume that σ_+ is co-prime with N, then σ_+^{-1} exists and

$$\mathrm{Dec}\left(\sigma_+^{-1}\otimes\mathrm{Enc}\left(\left(\Upsilon * \widehat{\Gamma_+}\right)[i,j]\right)\right) = \left(\Upsilon * \Gamma_+\right)[i,j]\,. \tag{78}$$

It seems plausible that this idea could be developed on top of batching and therefore allow a factor of σ_+ to be eliminated thereby allowing a larger $\hat{m}$ as defined in (75).

Compliance with Ethical Standards

Funding: This technical data deliverable was developed using contract funds under Basic Contract No. W56KGU-18-D-0004.

Conflicts of interest: All authors certify that they have no conflicts to declare that are relevant to the content of this article.

References

[1] Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys **51**(4), 1–35 (2018)

[2] Badawi, A.A., Chao, J., Lin, J., Mun, C.F., Sim, J.J., Tan, B.H.M., Nan, X., Aung, K.M.M., Chandrasekhar, V.R.: The alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus. arXiv [cs.CR] (2019). Arxiv.org/abs/1811.00778

[3] Bianchi, T., Piva, A., Barni, M.: Encrypted domain dct based on homomorphic cryptosystems. EURASIP Journal on Information Security **2009**(716357), 1–12 (2009)

[4] Chao, J., Badawi, A.A., Unnikrishnan, B., Lin, J., Mun, C.F., Brown, J.M., Campbell, J.P., an Jayashree Kalpathy-Cramer, M.C., Chandrasekhar, V.R., Krishnaswamy, P., Aung, K.M.M.: Carenets: Compact and resource-efficient cnn for homomorphic inference on encrypted medical images. arXiv [cs.CR] (2019). Arxiv.org/abs/1901.10074

[5] Equifax Inc.: Equifax announces cybersecurity incident involving consumer information. investor.equifax.com/news-and-events/news/2017/09-07-2017-213000628 (2017)

[6] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012). eprint.iacr.org/2012/144

[7] Fu, W., Lin, R., Inge, D.: Fully homomorphic image processing. arXiv [cs.CR] (2018). Arxiv.org/abs/1810.03249

[8] Ge, T., Zdonik, S.: Answering aggregation queries in a secure system model. In: Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), pp. 519–530. ACM (2007)

[9] Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, CA, USA (2009). Crypto.stanford.edu/craig/craig-thesis.pdf

[10] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Proceedings of the Annual International Cryptology Conference (CRYPTO), pp. 112–131. Springer-Verlag (1997)

[11] Hesamifard, E., Takabiy, H., Ghasemi, M.: Cryptodl: Deep neural networks over encrypted data. arXiv [cs.CR] (2017). Arxiv.org/abs/1711.05189

[12] Hsu, C.Y., Lu, C.S., Pei, S.C.: Image feature extraction in encrypted domain with privacy-preserving sift. IEEE Transactions on Image Processing **21**(11), 4593–4607 (2012)

[13] Hurtado-Guarnizo, O., Duarte-Gonzalez, M.E.: Study of somewhat homomorphic encryption scheme for image erosion and dilation. In: Proceedings Colombian Conference on Communications and Computing (COLCOM), pp. 212–217. IEEE (2018). In Spanish

[14] Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the ACM SIGSAC Conference on Computer and Communication Security (CSS), pp. 1209–1222. ACM (2018)

[15] Li, D., Dong, X., Cao, Z., Wang, H.: Privacy-preserving outsourced image feature extraction. Journal of Information Security and Applications **47**, 59–64 (2019)

[16] Li, L., Lu, R., Choo, K.K.R., Datta, A., Shao, J.: Privacy-preserving-outsourced association rule mining on vertically partitioned databases. IEEE Transactions on Information Forensics and Security **11**(8), 1847–1861 (2016)

[17] Mohanty, M., Rizwan, M., Russello, G.: 2dcrypt: Image scaling and cropping in encrypted domains. IEEE Transactions on Information Forensics and Security **11**(11), 2542–2555 (2016)

[18] N1 Analytics: Python-paillier (v1.4.0). GitHub (2018). Github.com/n1analytics/python-paillier

[19] Nagabhushana, S.: Computer Vision: Models, Learning, and Inference, 1 edn. New Age International (P) Limited, Publishers, 4835/24, Ansari Road, Daryaganj, New Delhi, 110002, INDIA (2005)

[20] Pailler, P.: Public-key cryptosystems based on composite degree residuosity classes. Lecture Notes in Computer Science **1592**, 223–238 (1999)

[21] Prince, S.: Computer Vision: Models, Learning, and Inference, 1 edn. Cambridge University Press, One Liberty Plaza, 20th Floor, New York, NY 10006, USA (2012)

[22] Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: R. Demillo, D. Dobkin, A. Jones, R. Lipton (eds.) Foundations of Secure Computation, pp. 171–181. Academic Press Inc., Orlando, FL, USA (1978)

[23] Schneider, M., Schneider, T.: Notes on non-interactive secure comparison in 'image feature extraction in the encrypted domain with privacy-preserving sift'. In: Proceedings of the ACM SIGSAC Conference on Computer and Communication Security (CSS), pp. 1209–1222. ACM (2018)

[24] Microsoft SEAL (release 3.0). www.sealcrypto.org (2018). Microsoft Research, Redmond, WA.

[25] United States Office of Personnel Management: Cybersecurity resource center: Cybersecurity incidents. www.opm.gov/cybersecurity/cybersecurity-incidents (2015)

[26] Vengadapurvaja, A., Nisha, G., Aarthy, R., Sasikaladevi, N.: An efficient homomorphic medical image encryption algorithm for cloud storage security. Procedia Computer Science **115**, 643–650 (2017)

[27] Wang, B., Zhan, Y., Zhang, Z.: Cryptanalysis of a symmetric fully homomorphic encryption scheme. IEEE Transactions on Information Forensics and Security **13**(6), 1460–1467 (2018)

[28] Wang, Q., Gao, L., Wang, H., Wei, X.: Face detection for privacy-protected images. IEEE Access **7**, 3918–3927 (2019)

[29] Yang, H., Huang, Y., Yu, Y., Yao, M., Zhang, X.: Privacy-preserving extraction of hog features based on integer vector homomorphic encryption. Lecture Notes in Computer Science **10701**, 102–117 (2017)

[30] Yang, P., Gui, X., An, J., Tian, F.: An efficient secret key homomorphic encryption used in image processing service. Security and Communication Networks **2017**, 1–11 (2017)

[31] Yang, T., Ma, J., Wang, Q., Miao, Y., Wang, X., Meng, Q.: Image feature extraction in encrypted domain with privacy-preserving hahn moments. IEEE Access **6**, 47,521–47,534 (2018)

[32] Zheng, P., Huang, J.: Discrete wavelet transform and data expansion reduction in homomorphic encrypted domain. IEEE Transactions on Image Processing **22**(6), 2455–2467 (2013)

[33] Ziad, M.T.I., Alanwar, A., Alzantot, M., Srivastava, M.: Cryptoimg: Privacy preserving processing over encrypted images. In: Proceedings 2nd Workshop on Security and Privacy in the Cloud (SPC 2016), pp. 570–575. IEEE (2016)