# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| 10/19/2020 | Technical Report | |

**4. TITLE AND SUBTITLE**
Pixel-Batched Homomorphic Encryption for Secure Outsourced Image Convolution

**5a. CONTRACT NUMBER**
W56KGU-19-D-0004

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**
Chris Giannella; James H Tanis; Adrian V Mariano

**5d. PROJECT NUMBER**
0719P04Z6E

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102

**8. PERFORMING ORGANIZATION REPORT NUMBER**
PRS-19-3093

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
United States Army (ACC-APG)

**10. SPONSOR/MONITOR'S ACRONYM(S)**
ACC-APG

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
Secure outsourced image processing requires a workflow where a data owner wants to simultaneously keep her imagery private while providing it to another party for storage and computation. A key challenge involves allowing the other party to perform meaningful computations on behalf of the data owner. Over the last ten years, homomorphic encryption research addressing this challenge has blossomed. Owing to the fundamental nature of convolution in image processing, we address the problem of secure outsourced image convolution. We start by developing an approach based on a straight-forward application of the Paillier cryptosystem where each pixel is encrypted separately. We improve upon this approach by batchi

**15. SUBJECT TERMS**
Image Processing; Homomorphic Encryption; Image Processing; Image Convolution;

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Susan Carpenito |
| | | | | 18 | 19b. TELEPHONE NUMBER *(Include area code)* 781-271-7646 |

# Pixel-Batched Homomorphic Encryption for Secure Outsourced Image Convolution

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

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

**Abstract**

Secure outsourced image processing requires a workflow where a data owner wants to simultaneously keep her imagery private while providing it to another party for storage and computation. A key challenge involves allowing the other party to perform meaningful computations on behalf of the data owner. Over the last ten years, homomorphic encryption research addressing this challenge has blossomed. Owing to the fundamental nature of convolution in image processing, we address the problem of secure outsourced image convolution. We start by developing an approach based on a straight-forward application of the Paillier cryptosystem where each pixel is encrypted separately. We improve upon this approach by batching vectors of pixels before encryption (an idea applied by others to different image processing algorithms). We carried out a series of experiments and observed 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 cryptographic systems, such as RSA, did not allow interesting functions to be computed on encrypted data, but doing so would be worthwhile and possible [1]. In the last ten years, homomorphic encryption research has blossomed, motivated in part by the following two scenarios.

*Scenario 1:* To reduce costs, many organizations rent computing infrastructure to store and perform computations on their data. However, the data owner may not fully trust the storage/computation provider and wish to encrypt her data before giving it to the provider. Homomorphic encryption provides a way for the provider to carry out computations without seeing the unencrypted data.

*Corresponding author, cgiannella@mitre.org Approved for public release, NGA Case 20-752.

*Scenario 2:* 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 [2]. 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 [3]. One strategy to mitigate the damage in cases like these is for the data to be kept in encrypted form for as much of its lifetime as feasible. Under this strategy, the data owner and the computation provider are parts of the same organization. The data owner can encrypt and decrypt the data while the computation provider cannot decrypt. The goal is for the data owner to decrypt data as infrequently as possible. Homomorphic encryption provides a way to address this 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 [4].

Figure 1 illustrates the workflow for homomorphic encryption that we adopt which encompases both scenarios. The data owner wants to compute a function $f$ on her data but lacks the resources and wishes to keep the data private. The computation provider, on the other hand, has the computational resources but lacks access to the data owner's (private) data. The data owner encrypts her data and sends it to the computation provider who computes a function $f^*$ on the encrypted data. The provider sends the result back to the data owner who decrypts it (sometimes also performs a minor computation on the decrypted result) and recovers the results of computing $f$ on the original data. We refer to this workflow as *secure outsourced computation*.

Different workflows have been proposed and homomorphic encryption approaches developed for them. For example, other workflows require a third computation provider [5] or multiple rounds of communication between data owner and computation provider. These kinds of workflows are beyond the scope of this paper.

### *1.1. Homomorphic Encryption: Overview*

Many cryptosystems have been developed allowing operations to be performed homomorphically on encrypted data. A good survey was written by Acar *et al.* [6]. 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, either addition or multiplication, to be performed homomorphically on pairs of ciphertexts an unlimited number of times.[3] Details regarding one of these systems, the Paillier cryptosystem [7], are provided in Subsection 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

[3]Homomorphic operations between pairs of ciphertexts and plaintexts are also allowed.

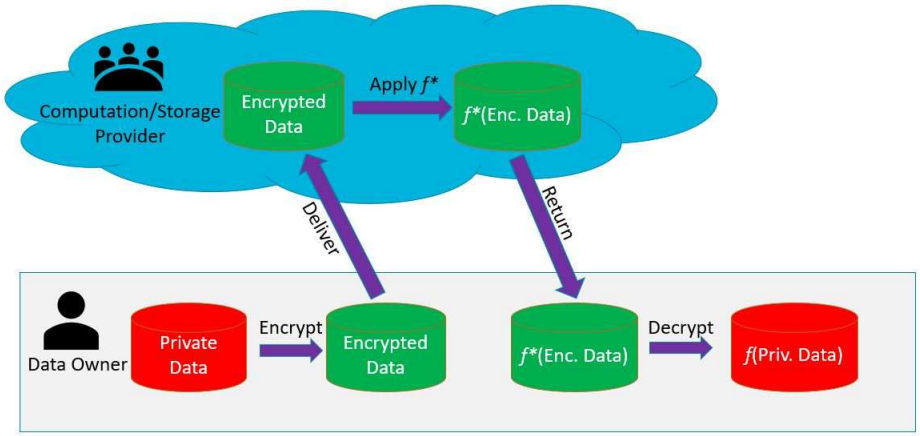


Figure 1: Secure outsourced computation

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)* [8]. He did this by modifying an earlier PHE scheme by Goldreich *et al.* [9] 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). Many papers have been published regarding the application of PHE and LFHE systems to the development of outsourced image processing algorithms - a summary is provided in Section 2. In this paper, we focus on applying one common PHE system, Paillier, to outsourced image processing, specifically, image convolution.

## 2. Related Work: Privacy-Preserving Outsourced 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): [10], [11], [12], [13]. A primary challenge faced by these efforts involves polynomial approximations of 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: [14], [15], [16]. Finally, yet other efforts have focused on implementing a variety of low-level image processing algorithms, *e.g.* color enhancement, image scaling, image JPEG compression/decompression: [17], [18], [19].

### *2.2. Partially Homomorphic Encryption Schemes*

Compared to LFHE systems, PHE systems are simplier to use when implementing image processing algorithms 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 [20], the discrete cosine transformation [21], discrete wavelet transformations [22], and convolution [23].

In an effort keep the computational power of LFHE systems without the difficulty of noise, some researchers have attempted to develop methods for implementing 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.* [14] claimed to have an implementation of comparison on encrypted data utilizing Paillier. On top of this, they implemented SIFT feature extraction on encrypted data. However, Schneider [24] argued that Hsu's approach is not secure. Separately, Li *et al.* [25] claimed to have an implementation of comparison which was utilized in [26] for outsourced medical image analysis and in [27] for secure face matching. However, Wang *et al.* [28] argued that Li's approach is insecure.

#### *2.2.1. Most Relevant*

Ziad *et al.* [23] 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.* [21] 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.* [20] 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 [4]. 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.

## 3. Preliminaries

Let $\mathbb{Z}_{>0}$ denote the positive integers. Given $m \in \mathbb{Z}_{>0}$, let $\mathbb{Z}_m$ denote the set $\{z \in \mathbb{Z} : 0 \leq z < m\}$ and let $\mathbb{Z}_m^*$ denote the set $\{z \in \mathbb{Z}_{>0} : z < m \text{ and } z \text{ is co-prime with } m\}$. Given $z \in \mathbb{Z}_m^*$, $z^{-1}$ denotes the multiplicative inverse modulo $m$, which is guaranteed to exist since $z$ is co-prime with $m$.

We sometimes apply to matrices operations that take a number as input. This shorthand should be understood in the natural way - the operation is applied element-wise to the matrix. We use Python-style indexing notation to denote sub-matrices. Let $M$ denote an $n \times n$ 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 $M[a_1, b_1 : b_2]$ ($M[a_1 : a_2, b1]$) 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 $M[a_2 - 1, b_1]$). We use : to denote $0 : n$.

### *3.1. Image Convolution*

Convolution is a commonly employed linear filtering operation [29] §13.1.3. Let $\Upsilon$ denote an $n \times n$ image (a matrix) where each pixel assumes a greyscale value in $\{0, 1, 2, \cdots, 255\}$. Let $\Gamma$ denote a $k \times k$ kernel matrix ($k < n$ 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 $(n - k + 1) \times (n - k + 1)$ matrix whose $(i, j)$ entry, $0 \leq i, j \leq n - k$, is

$$\sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \Upsilon[i + a, j + b]\Gamma[a, b].$$

We computed convolution using a shift-add algorithm[4] (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 $\Upsilon$ by each kernel entry regardless of the fact that some of the entries may have the same value. We used the dictionary KerMap to avoid this redundancy by recording

[4]The non-optimized version of shift-add is sketched in [30] §3.2.1.

for each *unique* kernel entry value, the multiplication of $\Upsilon$ by that value.

**Data:** $\Upsilon$,$\Gamma$
**Result:** $C$, the convolution of $\Upsilon$ and $\Gamma$
KerMap $\leftarrow \{\}$; $C \leftarrow (n-k+1) \times (n-k+1)$ 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$[\Gamma[a,b]][a : a+n-k+1, b : b+n-k+1]$
  **end**
**end**

**Algorithm 1:** Shift-add convolution with an optimization.

### *3.2. The Paillier Cryptosystem*

The security of the Paillier cryptosystem is based on the conjectured computational difficulty of solving the composite residuosity problem discussed in [7]. 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 $\lambda$ denote the least common multiple of $p-1$ and $q-1$.

- The encryption key is $(N, g)$ with $g$ drawn uniformly from $\left\{\hat{g} \in \mathbb{Z}^*_{N^2} : \left\lfloor \frac{\hat{g}^\lambda \pmod{N^2}-1}{N} \right\rfloor \in \mathbb{Z}^*_N \right\}$.
- The decryption key is $(\lambda, \mu)$ with $\mu \stackrel{\text{def}}{=\!=} \left\lfloor \frac{g^\lambda \pmod{N^2}-1}{N} \right\rfloor^{-1}$.

*Encryption:* Given plaintext $z$, select $r$ uniformly from $\left\{\hat{r} \in \mathbb{Z}^*_{N^2} : \hat{r} < N\right\}$, then compute $\text{Enc}(z) \stackrel{\text{def}}{=\!=} g^z r^N \pmod{N^2}$.

*Decryption:* Given ciphertext $\hat{z}$, compute $\text{Dec}(\hat{z}) \stackrel{\text{def}}{=\!=} \left\lfloor \frac{\hat{z}^\lambda \pmod{N^2}}{N} \right\rfloor \mu \pmod{N}$.

*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 $z_1, z_2$:

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

- $\mathrm{Dec}\left(\mathrm{Enc}\left(z_1\right) \otimes z\right) = z_1 z \pmod{N}$.

Unlike LHE 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 Private Outsourced 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 $\Gamma_+$ and $\Gamma_-$, both $k \times k$ matrices:

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

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

- $\sigma_+ \stackrel{\text{def}}{=\!=} \min_{\sigma \in \mathbb{Z}_{>0}} \left\{ \max_{0 \le a,b < k} \left\{ \left| \frac{\lceil \sigma \Gamma_+[a,b] \rceil}{\sigma} - \Gamma_+[a,b] \right| \right\} \le \epsilon \right\}$.
- $\widehat{\Gamma_+}[a,b] \stackrel{\text{def}}{=\!=} \lceil \sigma_+ \Gamma_+[a,b] \rceil$ for each $a, b \in \{0, \ldots, k-1\}$.

We define $\widehat{\Gamma_-}$ and $\sigma_-$ in similar fashion and approximate $\Upsilon * \Gamma$ by $\overline{\Upsilon * \Gamma} \stackrel{\text{def}}{=\!=} \frac{\widehat{\Gamma_+} * \Upsilon}{\sigma_+} - \frac{\widehat{\Gamma_-} * \Upsilon}{\sigma_-}$.

### *4.1. Overall Approach*

The data owner computes $\mathrm{Enc}\left(\Upsilon\right)$, the $n \times n$ matrix of ciphertexts. The computation provider selects $\epsilon$ and computes $\widehat{\Gamma_+}, \sigma_+, \widehat{\Gamma_-}, \sigma_-$ as described earlier. Next, the computation provider computes $M_+$ and $M_-$ by applying Algorithm 2 to $\mathrm{Enc}\left(\Upsilon\right)$, $\widehat{\Gamma_+}$ and $\mathrm{Enc}\left(\Upsilon\right)$, $\widehat{\Gamma_-}$, respectively. Finally, the data owner computes

$$\frac{\mathrm{Dec}\left(M_+\right)}{\sigma_+} - \frac{\mathrm{Dec}\left(M_-\right)}{\sigma_-} = \overline{\Upsilon * \Gamma}.$$

**Data:** $\mathrm{Enc}(\Upsilon)$ and $\widehat{\Gamma_{\pm}}$ a $k \times k$ non-negative integer matrix.
**Result:** $M_{\pm}$ a $(n-k+1) \times (n-k+1)$ matrix of ciphertexts.
KerMap $\leftarrow \{\}$; for each $(i,j)$, compute $\mathrm{Enc}(0)$ and assign it 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**
      $\mathrm{KerMap}[\widehat{\Gamma_{\pm}}[a,b]] \leftarrow \widehat{\Gamma_{\pm}}[a,b] \otimes \mathrm{Enc}(\Upsilon)$
    **end**
  **end**
**end**
**for** $a = 0 \cdots k-1$ **do**
  **for** $b = 0 \cdots k-1$ **do**
    $M_{\pm} \leftarrow M_{\pm} \oplus \mathrm{KerMap}[\widehat{\Gamma_{\pm}}[a,b]][a : a+n-k+1, b : b+n-k+1]$
  **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 $\Upsilon$ 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 $(p_0, \cdots, p_{m-1})$ using a single plaintext. We define[5] a base $B$ number, $\mathrm{Bat}_B(p_0, \cdots, p_{m-1})$, whose digits are $p_0, \cdots, p_{m-1}$.

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

To recover the vector, we perform a series of modulos and divisions by $B$ to extract the digits from $\mathrm{Bat}_B(p_0, \cdots, p_{m-1})$ and denote this operation $\mathrm{UBat}_B(.)$.

To see how convolution is performed on top of batching, let $\mathrm{Bat}_B(\Upsilon)$ be the length $n$ column vector with $i^{th}$ entry $\mathrm{Bat}_B(\Upsilon[i,:])$. Let $\mathrm{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)$ denote the $n \times k$ matrix whose $c^{th}$ column $(0 \le c < k)$ is $\widehat{\Gamma_+}[a,b] B^{k-1-c} \mathrm{Bat}_B(\Upsilon)$.

### 5.1. Illustration Case: $k = 3$ and $n = 4$

Given $0 \le i < 4$ and $0 \le a < 3$, $\sum_{b=0}^{2} \mathrm{KM}_B\left(\widehat{\Gamma_+}[a, 2-b]\right)[i, 2-b]$ equals

$$\mathrm{KM}_B\left(\widehat{\Gamma_+}[a,2]\right)[i,2] + \mathrm{KM}_B\left(\widehat{\Gamma_+}[a,1]\right)[i,1] + \mathrm{KM}_B\left(\widehat{\Gamma_+}[a,0]\right)[i,0],$$

which equals

[5] The base is chosen to be sufficiently large as discussed later.

$$\widehat{\Gamma_+}[a,2]B^0\mathrm{Bat}_B\left(\Upsilon\right)[i]+\widehat{\Gamma_+}[a,1]B^1\mathrm{Bat}_B\left(\Upsilon\right)[i]+\widehat{\Gamma_+}[a,0]B^2\mathrm{Bat}_B\left(\Upsilon\right)[i],$$

which equals

$$\left(\sum_{j=0}^{3}\widehat{\Gamma_+}[a,2]\Upsilon[i,j]B^j\right)+\left(\sum_{j=0}^{3}\widehat{\Gamma_+}[a,1]\Upsilon[i,j]B^{j+1}\right)+\left(\sum_{j=0}^{3}\widehat{\Gamma_+}[a,0]\Upsilon[i,j]B^{j+2}\right),$$

which equals[6]

$$\begin{array}{rl} & \left(\ldots\right)B^0+\left(\ldots\right)B^1 \\ + & \left(\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[i,b]\right)B^2+\left(\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[i,b+1]\right)B^3 \\ + & \left(\ldots\right)B^4+\left(\ldots\right)B^5. \end{array}$$

Let $\mathrm{RowBConv}_{+,3,4}^{(a)}$ denote[7] the length four column vector with $i^{th}$ entry $\sum_{b=0}^{2}\mathrm{KM}_B\left(\widehat{\Gamma_+}[a,2-b]\right)[i,2-b]$. Then, $\sum_{a=0}^{2}\mathrm{RowBConv}_{+,3,4}^{(a)}[a]$ equals

$$\begin{array}{rl} & \left(\ldots\right)B^0+\left(\ldots\right)B^1 \\ + & \left(\sum_{a=0}^{2}\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[a,b]\right)B^2+\left(\sum_{a=0}^{2}\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[a,b+1]\right)B^3 \\ + & \left(\ldots\right)B^4+\left(\ldots\right)B^5. \end{array}$$

The coefficients on $B^2$ and $B^3$ are $\left(\Upsilon*\widehat{\Gamma_+}\right)[0,0]$ and $\left(\Upsilon*\widehat{\Gamma_+}\right)[0,1]$, respectively. Likewise, $\sum_{a=0}^{2}\mathrm{RowBConv}_{+,3,4}^{(a)}[a+1]$ equals

$$\begin{array}{rl} & \left(\ldots\right)B^0+\left(\ldots\right)B^1 \\ + & \left(\sum_{a=0}^{2}\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[a+1,b]\right)B^2+\left(\sum_{a=0}^{2}\sum_{b=0}^{2}\widehat{\Gamma_+}[a,b]\Upsilon[a+1,b+1]\right)B^3 \\ + & \left(\ldots\right)B^4+\left(\ldots\right)B^5. \end{array}$$

The coefficients on $B^2$ and $B^3$ are $\left(\Upsilon*\widehat{\Gamma_+}\right)[1,0]$ and $\left(\Upsilon*\widehat{\Gamma_+}\right)[1,1]$, respectively. Define length two column vector

[6]The coefficients on $B^0$, $B^1$, $B^4$, and $B^5$ are unimportant hence are not shown.

[7]The last two subscripts on $\mathrm{RowBConv}_{+,3,4}$ stem from the values of $k$ and $n$, respectively.

$$\mathrm{BConv}_{+,3,4} \stackrel{\text{def}}{=\!=} \begin{bmatrix} \sum_{a=0}^{2} \mathrm{RowBConv}^{(a)}_{+,3,4}\,[a] \\ \sum_{a=0}^{2} \mathrm{RowBConv}^{(a)}_{+,3,4}\,[a+1] \end{bmatrix}.$$

To extract $\Upsilon * \widehat{\Gamma_+}$ from $\mathrm{BConv}_{+,3,4}$, each entry in the convolution must be less than $B$. This is because the entries of the convolution are digits in base $B$ numbers. Hence, we assume $255 \sum_{a=0}^{2} \sum_{b=0}^{2} \widehat{\Gamma_+}[a,b] < B$. Under this assumption, it follows that

- $\mathrm{UBat}_B\left(\mathrm{BConv}_{+,3,4}\right)[:, 2:4] = \Upsilon * \widehat{\Gamma_+}$
- $\max_{i=0}^{1}\{\mathrm{BConv}_{+,3,4}[i]\} < B \sum_{j=0}^{3} B^j$

where $\mathrm{UBat}_B(.)$ is applied separately to each entry of $\mathrm{BConv}_{+,3,4}$ producing a row in the final result.

*5.2. The General Case: Dropping the Assumptions $k = 3$ and $n = 4$*

Without loss of generality[8], we develop this case only for $\Gamma_+$ and $\mathrm{BConv}_{+,k,n}$. We have: if $255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b] < B$, then

- $\mathrm{UBat}_B\left(\mathrm{BConv}_{+,k,n}\right)[:, k-1:n] = \Upsilon * \widehat{\Gamma_+}$ (1)
- $\max_{i=0}^{n-k}\{\mathrm{BConv}_{+,k,n}[i]\} < B \sum_{j=0}^{n-1} B^j$. (2)

These results form the basis of the algorithm for carrying out convolution on top of batching. To this end, we express $\mathrm{BConv}_{+,k,n}$ in a form more conducive to developing the algorithm (recall $\mathrm{BConv}_{+,k,n}$ is a length $n-k+1$ column vector), namely, a sum of column vectors. First observe that:

$$\begin{aligned}
\mathrm{BConv}_{+,k,n}[i] &= \sum_{a=0}^{k-1} \mathrm{RowBConv}^{(a)}_{+,k,n}\,[a+i] && (3)\\
&= \sum_{a=0}^{k-1} \left( \sum_{b=0}^{k-1} \mathrm{KM}_B\left(\widehat{\Gamma_+}[a, k-1-b]\right)[a+i, k-1-b] \right) && (4)\\
&= \sum_{a=0}^{k-1} \left( \sum_{b=0}^{k-1} \mathrm{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)[a+i, b] \right) && (5)
\end{aligned}$$

where (3) and (4) follow from plugging in definitions and (5) follows from changing the index on the inner summation. Hence,

[8] For $\Gamma_-$ and $\mathrm{BConv}_{-,k,n}$, replace $+$'s in subscripts with $-$'s.

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

Motivated by the definition of $\mathrm{KM}_B\left(\widehat{\Gamma_+}[a,b]\right)$, let $\mathrm{EKM}_B\left(\widehat{\Gamma_+}[a,b]\right)$ denote the $n \times k$ matrix whose $c^{th}$ column $(0 \le c < k)$ is $\left(\widehat{\Gamma_+}[a,b]B^{k-1-c}\right) \otimes \mathrm{Enc}\left(\mathrm{Bat}_B\left(\Upsilon\right)\right)$. Let

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

If each entry of $\mathrm{BConv}_{+,k,n}$ is less than $N$, this vector is recovered by decryption, namely, $\mathrm{Dec}\left(\mathrm{EBConv}_{+,k,n}\right) = \mathrm{BConv}_{+,k,n}$. Furthermore, if $255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b] < B$, then (1) implies that

$$\mathrm{UBat}_B\left(\mathrm{Dec}\left(\mathrm{EBConv}_{+,k,n}\right)\right)[:, k-1:n] = \Upsilon * \widehat{\Gamma_+}. \tag{6}$$

*5.3. Overall Approach*

The data owner sets

$$B = \max\left\{255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_+}[a,b], 255 \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} \widehat{\Gamma_-}[a,b]\right\} + 1 \tag{7}$$

The data owner computes $\mathrm{Enc}\left(\mathrm{Bat}_B\left(\Upsilon\right)\right)$, the length $n$ vector of ciphertexts. The computation provider selects $\epsilon$ and computes $\widehat{\Gamma_+}, \sigma_+, \widehat{\Gamma_-}, \sigma_-$ as described earlier. Next, the computation provider computes $\mathrm{EBC}_+$ and $\mathrm{EBC}_-$ by applying Algorithm 3 to $\mathrm{Enc}\left(\mathrm{Bat}_B\left(\Upsilon\right)\right)$, $\widehat{\Gamma_+}$ and $\mathrm{Enc}\left(\mathrm{Bat}_B\left(\Upsilon\right)\right)$, $\widehat{\Gamma_-}$, respectively. Finally, the data owner computes

$$\frac{\mathrm{UBat}_B\left(\mathrm{Dec}\left(\mathrm{EBC}_+\right)\right)[:, k-1:n]}{\sigma_+} - \frac{\mathrm{UBat}_B\left(\mathrm{Dec}\left(\mathrm{EBC}_-\right)\right)[:, k-1:n]}{\sigma_-}. \tag{8}$$

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

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

**end**

**Algorithm 3:** Shift-add convolution in batched space.

*5.3.1. Correctness Proof*
The proof is contingent upon the assumption

$$B\sum_{j=0}^{n-1} B^j < N. \tag{9}$$

It can be seen that $\mathrm{EBC}_+$ equals $\mathrm{EBConv}_{+,k,n}$ and $\mathrm{EBC}_-$ equals $\mathrm{EBConv}_{-,k,n}$. Inequality (2) implies that each entry of $\mathrm{BConv}_{+,k,n}$ and $\mathrm{BConv}_{-,k,n}$ are less than $N$. Therefore, (6) implies that (8) equals

$$\frac{\Upsilon * \widehat{\Gamma_+}}{\sigma_+} - \frac{\Upsilon * \widehat{\Gamma_-}}{\sigma_-} = \overline{\Upsilon * \Gamma}$$

as needed.

*5.3.2. Addressing the Constraint on n*
The correctness proof requires $N$, $B$, and $n$ to satisfy inequality (9). However, $B$ is fixed as set by equation (7), and $N$ is also fixed as it is typically set to a value sufficient for security. Let $\hat{n}$ denote the largest integer which satisfies inequality (9) when put in the place of $n$. The data owner creates separate[9]

[9]The $\Upsilon'_\ell s$ overlap to accommodate edges being ignored in convolution.

images $\Upsilon_0, \cdots, \Upsilon_r$ where $\Upsilon_0$ contains the first $\hat{n}$ columns; $\Upsilon_1$ contains columns $\hat{n} - \frac{k-1}{2}$, $\hat{n} - \frac{k-1}{2} + 1$, $\cdots$, $2\hat{n} - \frac{k-1}{2}$; *etc.*. Then, the data owner initiates the overall approach above separately on each $\Upsilon_\ell$ and concatenates the final results to produce $\overline{\Upsilon * \Gamma}$.

## 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 [31]. 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[10] a $512 \times 512$ greyscale image and six kernels: box blur kernels[11] of sizes 3, 5, and 7 and approximate Gaussian blur kernels of the same sizes - see Table 1. 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.

For each experiment, we carried out six trials during each of which we computed[12] the run-times of plaintext, non-batched Paillier, and batched Paillier convolution for a fixed kernel, fixed number of bits for $N$, and $\epsilon$ fixed[13] at 0.00001. 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 $\{512, 1024, 2048\}$.

### *6.1. Results*

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

[10] We used only one image since the run times do not depend upon the contents of the image.

[11] All entries of the kernel are the same and sum to one.

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

[13] Varying $\epsilon$ produced little change so it is fixed in all experiments we report.

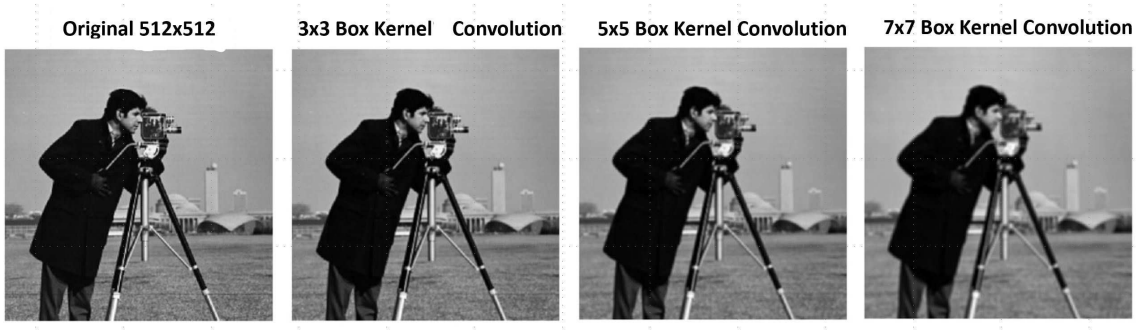


Figure 2: Image used in all experiments (left, ©Massachusetts Institute of Technology, available under `creativecommons.org/licenses/by-nc-sa/3.0`) and those produced by box blur.

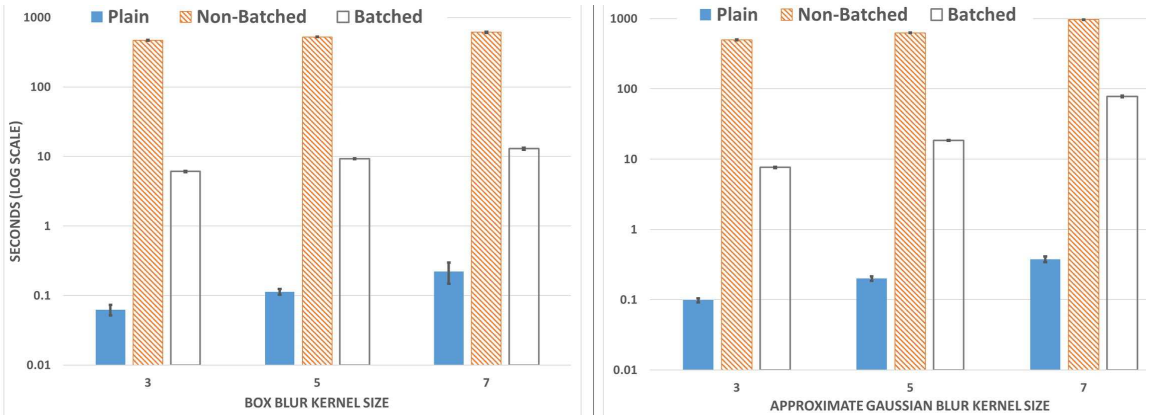


Figure 3: Experiments with varying kernel sizes and the bit length of $N$ fixed at 1024.

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 $\sigma_+$.

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.

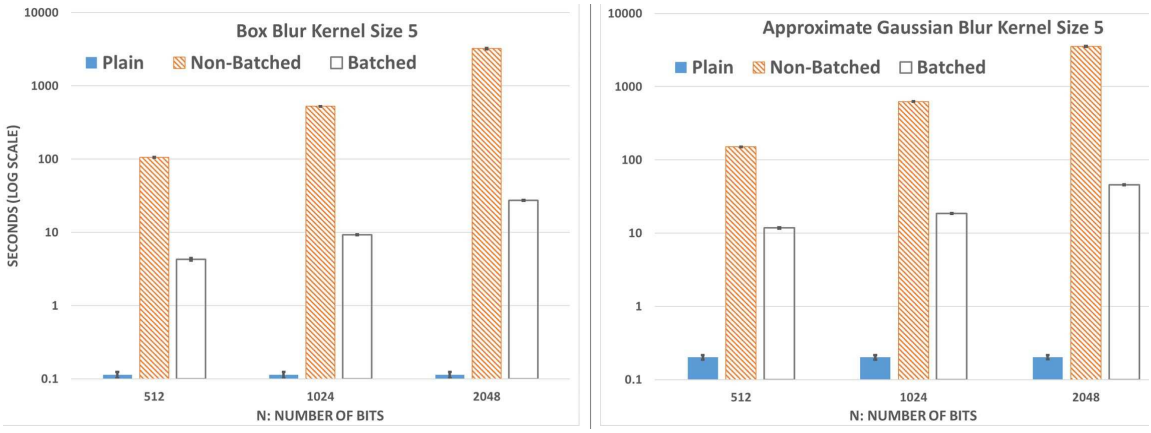


Figure 4: Experiments with varying bit length of $N$ and kernel sizes fixed at five.

## 7. Summary and Future Work

We addressed the problem of secure outsourced image convolution. We developed a straight-forward 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 [21] and bilinear scaling [20]. 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.* [22] 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 $\sigma_+$ is co-prime with $N$, then $\sigma_+^{-1}$ exists and

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

It seems plausible that this idea could be developed on top of batching and therefore allow a factor of $\sigma_+$ to be eliminated thereby allowing a larger $\hat{n}$.

## References

[1] R. Rivest, L. Adleman, M. Dertouzos, On data banks and privacy homomorphisms, in: R. Demillo, D. Dobkin, A. Jones, R. Lipton (Eds.), Foun-

dations of Secure Computation, Academic Press Inc., Orlando, FL, USA, 1978, pp. 171–181.

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

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

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

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

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

[7] P. Pailler, Public-key cryptosystems based on composite degree residuosity classes, Lecture Notes in Computer Science 1592 (1999).

[8] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Department of Computer Science, Stanford University, Stanford, CA, USA, https://crypto.stanford.edu/craig/craig-thesis.pdf (7 2009).

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

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

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

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

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

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

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

[16] T. Yang, J. Ma, Q. Wang, Y. Miao, X. Wang, Q. Meng, Image feature extraction in encrypted domain with privacy-preserving hahn moments, IEEE Access 6 (2018) 47521–47534.

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

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

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

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

[21] T. Bianchi, A. Piva, M. Barni, Encrypted domain dct based on homomorphic cryptosystems, EURASIP Journal on Information Security 2009 (716357) (2009).

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

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

[24] M. Schneider, T. Schneider, 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), ACM, 2018, pp. 1209–1222.

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

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

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

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

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

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

[31] N1 Analytics, Python-paillier (v1.4.0), GitHub, `https://github.com/n1analytics/python-paillier` (2018).