CryptoImg: Privacy Preserving Processing Over Encrypted Images

M. Tarek Ibn Ziad*, Amr AlAnwar†, Moustafa Alzantot*, and Mani Srivastava†
Motivation

- Cloud computing provide scalable solution for data storage and processing.

- Emerging solutions for image editing on the cloud: Adobe creative cloud, Pixlr, .., etc.

- Images usually contain privacy sensitive. Outsourcing the raw data exposes a lot of information.

- How to protect user’s privacy while editing images in the cloud?

Video courtesy of Ankita Lathey, P K. Atrey, Image Enhancement in Encrypted Domain over Cloud, ACM TOMM 2015
CryptoImg

Client Device

Cloud Server

Encrypt

Decrypt

CryptoImg Operation
## Related Work

<table>
<thead>
<tr>
<th>Work</th>
<th>Description</th>
<th>Comparison to CryptoImg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortell, et al.</td>
<td>Implements brightness / contrast filters using fully homomorphic encryption</td>
<td>- <strong>CryptoImg</strong> supports more operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>CryptoImg</strong> is more computationally efficient.</td>
</tr>
<tr>
<td>Lathey, and atrey, et al.</td>
<td>Image enhancement (e.g. spatial filtering, anti-aliasing, edge enhancement, etc.) using Shamir Secret sharing</td>
<td>- Security guarantees require distributing the image processing task across non-colluding servers.</td>
</tr>
<tr>
<td>Hu et al.</td>
<td>Secure linear filtering using SMC</td>
<td>- <strong>CryptoImg</strong> is more computationally efficient</td>
</tr>
</tbody>
</table>
Background

• Homomorphic encryption allows carrying out computations on ciphertext.

  • **Fully homomorphic encryption:**
    
    • Performs arbitrary computations.
    
    • Computationally expensive.

  • **Partially Homomorphic encryption:**
    
    • Supports either addition or multiplication of encrypted values.

• Paillier encryption is an additive homomorphic encryption scheme.

  • Supports the addition of two encrypted values.
  
  \[
  \text{DEC}([m_1] \oplus [m_2]) = \text{DEC}([m_1 \times m_2] \mod n^2) = (m_1 + m_2) \mod n
  \]

  • Can multiply an encrypted value by another plain scaler.

  \[
  \text{DEC}([m_1] \otimes d) = \text{DEC}([m_1]^d \mod n^2) = (m \times d) \mod n
  \]
Challenges

• Paillier is defined over the group of positive integers $\mathbb{Z}_n^*$.  

• In practice, we also need to deal with negative and real numbers.  

• Solution:
  
  • Use an encoding scheme that maps negative and real numbers to integers and preserves the Paillier encryption homomorphic properties.
Challenges

- Paillier is defined over the group of positive integers $\mathbb{Z}_n^*$. 
- In practice, we also need to deal with negative and real numbers.

**Solution:**

- Use an encoding scheme that maps negative and real numbers to integers and preserves the Paillier encryption homomorphic properties.
- To represent negative numbers. Assign different ranges for positive and negative values.
  - $[0, n/3]$ : Positive numbers
  - $[2n/3, n]$ : Negative numbers.
  - $[n/3, 2n/3]$ : Reserved for overflow detection
Challenges

- Paillier is defined over the group of positive integers $\mathbb{Z}_n^*$.

- In practice, we also need to deal with negative and real numbers.

**Solution:**

- Represent each real number by a pair $(m, e)$ where:
  - $m$: mantissa
  - $e$: non-negative exponent

- To encrypt a real number, it is sufficient to encrypt only the mantissa $m$. 
Challenges

- Paillier is defined over the group of positive integers $\mathbb{Z}_n^*$.

- In practice, we also need to deal with negative and real numbers.

- **Solution:**

<table>
<thead>
<tr>
<th>Protocol 1 Secure FP Numbers Processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiplication:</strong> $[c] = a \otimes [b]$</td>
</tr>
<tr>
<td>$[m_c] = m_a \otimes [m_b]$</td>
</tr>
<tr>
<td>$e_c = e_a + e_b$</td>
</tr>
<tr>
<td><strong>Addition:</strong> $[c] = [a] \oplus [b]$</td>
</tr>
<tr>
<td>if $e_a \leq e_b$</td>
</tr>
<tr>
<td>$[m_c] = [m_a] \oplus [Base^{e_b-e_a} \otimes [m_b]], e_c = e_a$</td>
</tr>
<tr>
<td>if $e_a &gt; e_b$</td>
</tr>
<tr>
<td>$[m_c] = [m_b] \oplus [Base^{e_a-e_b} \otimes [m_a]], e_c = e_b$</td>
</tr>
</tbody>
</table>
CryptoImg Operations

- **CryptoImg** supports the following image processing operations:
  - Image Adjustment
  - Noise Reduction
  - Edge Detection
  - Morphological Operations
  - Histogram Equalization
Image Adjustment

- Adding or subtracting adjustment value from each pixel.
- Client sends the encrypted Image $[I]$, and adjustment value $v$ to the server.
  $$[r] = [i] \oplus [v]$$
- Server applies the adjustment to each pixel.
- Client decrypts the result.
Image Noise Reduction

- Client sends the encrypted Image $[I]$, and the filter values $f$ to the server.

$$[I_{spf}(u,v)] = \frac{1}{m \times n} \otimes \sum_{u=1,v=1}^{m,n} f(u,v) \otimes [I(u,v)]$$

- Server computes the output image and sends it to the client.

- Client decrypts the result.
Edge Detection

- Client encrypts the source image $I$.

- Servers computes the encrypted horizontal and vertical gradients of image:
  
  $$ [G_x(u,v)] = \sum_{u=1,v=1}^{m,n} h_1(u,v) \otimes [I(u,v)] $$
  
  $$ [G_y(u,v)] = \sum_{u=1,v=1}^{m,n} h_2(u,v) \otimes [I(u,v)] $$

- Client decrypts the result to compute the gradient magnitude and direction:
  
  $$ G = \sqrt{G_x^2 + G_y^2} $$
  
  $$ \Theta = \text{atan2} \left(G_y, G_x\right)$$
Morphological Operations

- Client encrypts the source image \( I \).
- Servers computes, and sends it to client.

\[
\begin{align*}
[L(u, v)] &= \sum_{u=1}^{m} \sum_{v=1}^{n} [I(u, v)]
\end{align*}
\]

- Client decrypts \( L \) and applies a threshold \( T \) to get the output image.
Histogram Equalization

- Client computes and encrypts the image histogram $[H]$. 

- Server computes the brightness transformation $[T(p)]$

\[
\begin{align*}
[H_c(0)] & = [H(0)] \\
[H_c(p)] & = [H_c(p-1)] \oplus [H(p)], \text{where } p = 1, 2, \ldots, G - 1 \\
[T(p)] & = (G - 1)/(w \times \ell) \otimes [H_c(p)].
\end{align*}
\]

- Server sends $[T(p)]$ to client.

- Client decrypts $T(p)$ and applies it to get the output image.
Evaluation

- **CryptolImg** is implemented as an extension for OpenCV library.
- Evaluated using both desktop and mobile device clients.
Evaluation

- **Cryptolmg** is implemented as an extension for OpenCV library.

- Evaluated using both desktop and mobile device clients.

Encryption/ Decryption Time (Sec) of 512x512 Image on Desktop Device

<table>
<thead>
<tr>
<th></th>
<th>512 bit</th>
<th>1024 bit</th>
<th>2048 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>156.905</td>
<td>1154.29</td>
<td>7670.49</td>
</tr>
<tr>
<td>Decryption</td>
<td>1.93</td>
<td>4.068</td>
<td>9.623</td>
</tr>
</tbody>
</table>

Encryption/ Decryption Time (Sec) of 256x256 Image on Mobile Device

<table>
<thead>
<tr>
<th></th>
<th>512 bit</th>
<th>1024 bit</th>
<th>2048 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>73</td>
<td>575</td>
<td>3701</td>
</tr>
<tr>
<td>Decryption</td>
<td>48</td>
<td>325</td>
<td>2268</td>
</tr>
</tbody>
</table>
Evaluation

- **CryptoImg** is implemented as an extension for *OpenCV* library.
- Evaluated using both desktop and mobile device clients.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Pre-Proc</th>
<th>Server Time</th>
<th>Post-Proc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PD</td>
<td>1024 bit</td>
<td>2048 bit</td>
</tr>
<tr>
<td></td>
<td>PC</td>
<td>Mob</td>
<td>PC</td>
</tr>
<tr>
<td>Brightness</td>
<td>0.00108</td>
<td>0.81</td>
<td>0</td>
</tr>
<tr>
<td>LPF</td>
<td>0.00763</td>
<td>180.50</td>
<td>609.199</td>
</tr>
<tr>
<td>Edge Detection</td>
<td>0.00642</td>
<td>147.56</td>
<td>482.195</td>
</tr>
<tr>
<td>Erosion</td>
<td>0.00009</td>
<td>4.049</td>
<td>10.808</td>
</tr>
<tr>
<td>Histogram Equalization</td>
<td>0.00174</td>
<td>0.177</td>
<td>0.0144</td>
</tr>
</tbody>
</table>
Future Work

- Speedup the encryption / decryption time of an image using hardware accelerators, e.g. GPUs.