CryptoImg: Privacy Preserving Processing Over Encrypted Images

M. Tarek Ibn Ziad[*], Amr AlAnwar[+], **Moustafa Alzantot[+]**, and Mani Srivastava[+]

AIN SHAMS UNIVERSITY

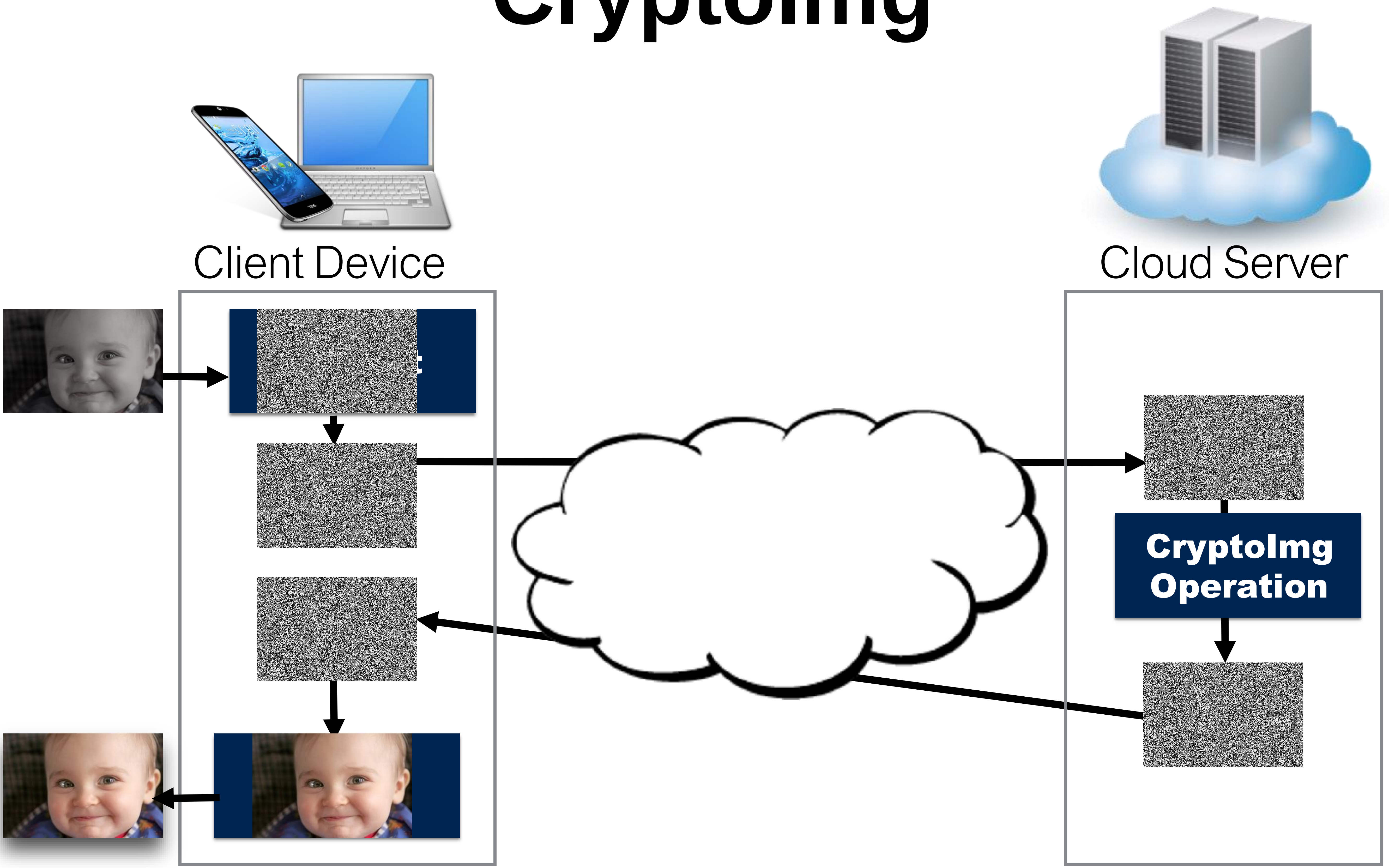
UCLA

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



Related Work

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

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.

$$\begin{aligned}\mathrm{DEC}([\![m_1]\!] \oplus_z [\![m_2]\!]) &= \mathrm{DEC}(([\![m_1]\!] \times [\![m_2]\!]) \, mod \, n^2) \\ &= (m_1 + m_2) \, mod \, n\end{aligned}$$

  - Can multiply an encrypted value by another plain scaler.

$$\begin{aligned}\mathrm{DEC}([\![m_1]\!] \otimes_z d) &= \mathrm{DEC}([\![m_1]\!]^d \, mod \, n^2) \\ &= (m \times d) \, mod \, n\end{aligned}$$

Challenges

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

Protocol 1 Secure FP Numbers Processing.

Multiplication: $[\![c]\!] = a \otimes [\![b]\!]$

$[\![m_c]\!] = m_a \otimes_z [\![m_b]\!]$

$e_c = e_a + e_b$

Addition: $[\![c]\!] = [\![a]\!] \oplus [\![b]\!]$

if $e_a \leq e_b$

$[\![m_c]\!] = [\![m_a]\!] \oplus_z (Base^{e_b - e_a} \otimes_z [\![m_b]\!]),\ e_c = e_a$

if $e_a > e_b$

$[\![m_c]\!] = [\![m_b]\!] \oplus_z (Base^{e_a - e_b} \otimes_z [\![m_a]\!]),\ e_c = e_b$

CryptoImg Operations

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

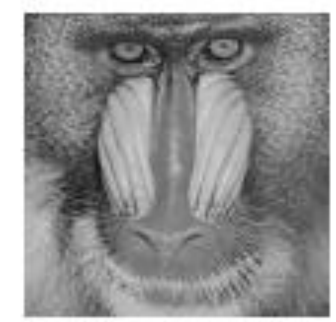
Image Adjustment

- Adding or subtracting adjustment value from each pixel.
- Client sends the encrypted Image **`[I]`**, and adjustment value **`v`** to the server.

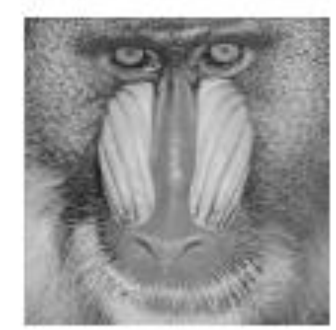
$$[\![r]\!] = [\![i]\!] \oplus [\![v]\!]$$

- Server applies the adjustment to each pixel.
- Client decrypts the result.

Brightness



Input



PD Output

Image Noise Reduction

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

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

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

- Client decrypts the result.

LPF

Input

PD Output

Edge Detection

- Client encrypts the source image **`I`**.
- Servers computes the encrypted horizontal and vertical gradients of imag

$$[\![G_x(u,v)]\!] = \sum_{u=1,v=1}^{m,n} h_1(u,v) \otimes [\![I(u,v)]\!]$$

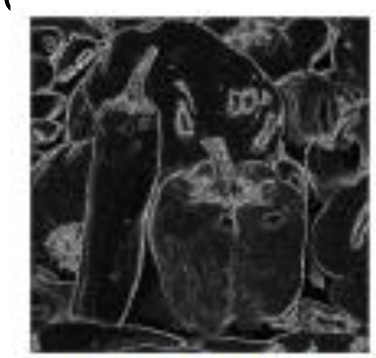
$$[\![G_y(u,v)]\!] = \sum_{u=1,v=1}^{m,n} h_2(u,v) \otimes [\![I(u,v)]\!]$$

- Client decrypts the result to compute the gradient magnitude and directi

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

$$\Theta = \operatorname{atan2}(G_y, G_x)$$

Input



PD Output

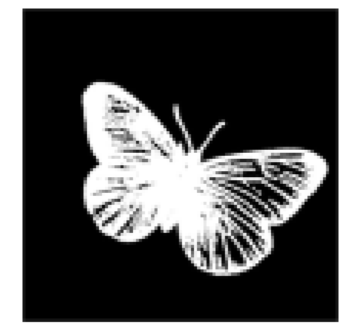
Morphological Operations

- Client encrypts the source image **`I`**.
- Servers computes, and sends it to client.

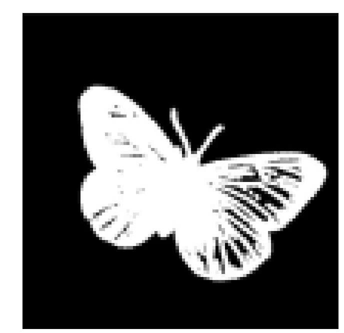
$$[\![L(u,v)]\!] = \sum_{u=1,v=1}^{m,n} [\![I(u,v)]\!].$$

- Client decrypts **`L`** and applies a threshold **`T`** to get the output image.

Dilation



Input



PD Output

Histogram Equalization

- Client computes and encrypts the image histogram **`[H]`**.
- Server computes the brightness transformation **`[T(p)]`**

$$
\begin{aligned}
&[\![H_c(0)]\!] = [\![H(0)]\!]\\
&[\![H_c(p)]\!] = [\![H_c(p-1)]\!] \oplus [\![H(p)]\!], \textit{where } p = 1, 2, \cdots G-1\\
&[\![T(p)]\!] = (G-1)/(w \times \ell) \otimes [\![H_c(p)]\!].
\end{aligned}
$$

- Server sends **`[T(p)]`** to client.
- Client decrypts **`T(p)`** and applies it to get the output image.

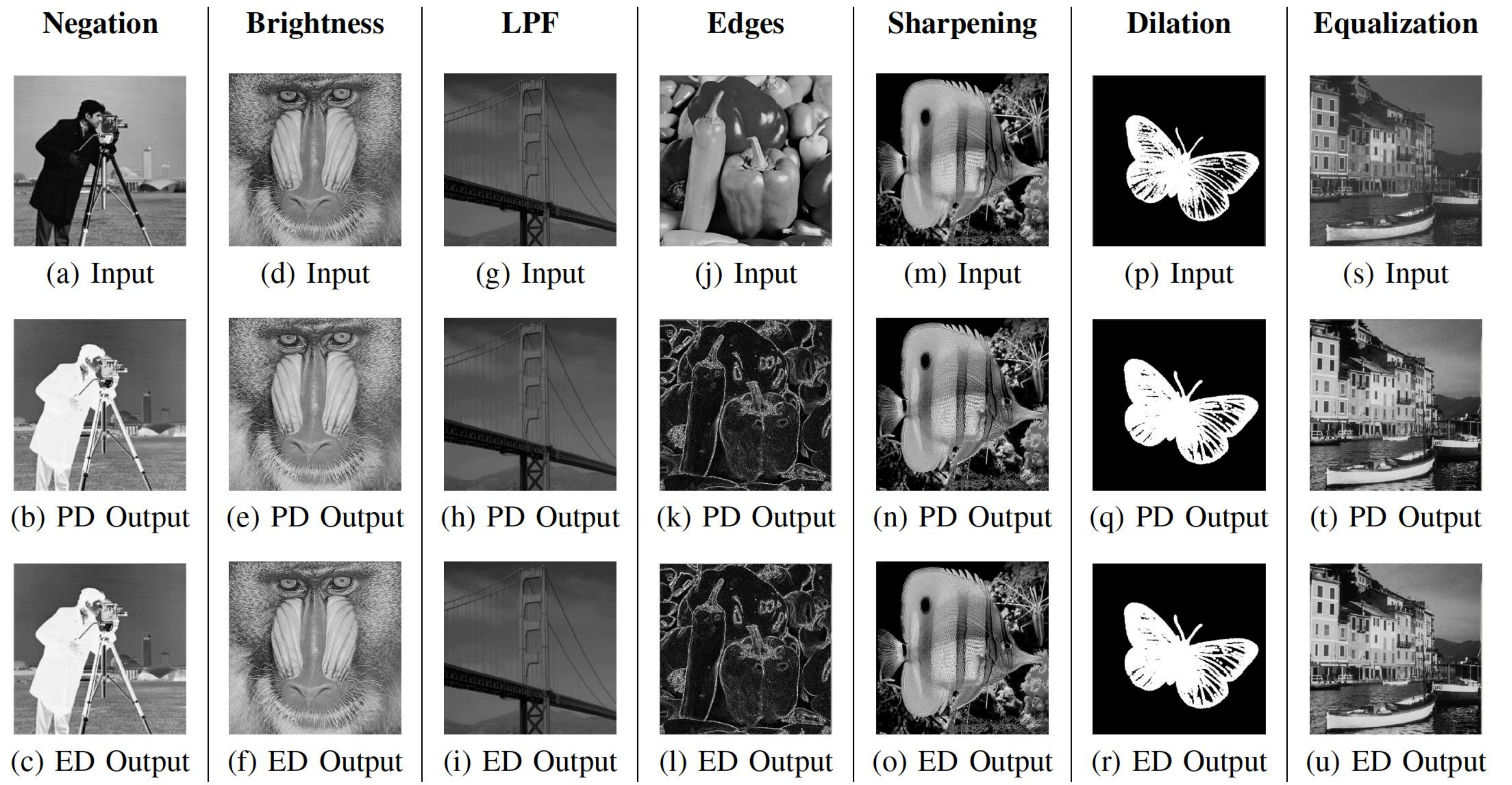
Equalization

Input

PD Output

Evaluation

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



Evaluation

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

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

	512 bit	1024 bit	2048 bit
Encryption	73	575	3701
Decryption	48	325	2268

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

	512 bit	1024 bit	2048 bit
Encryption	156.905	1154.29	7670.49
Decryption	1.93	4.068	9.623

Evaluation

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

Time for each operation (Sec.)

		Pre-Proc		Server Time		Post-Proc	
	PD	**PC**	**Mob**	**1024 bit**	**2048 bit**	**PC**	**Mob**
Brightness	0.00108	0	0	0.81	2.39	0	0
LPF	0.00763	0	0	180.50	609.199	0	0
Edge Detection	0.00642	0	0	147.56	482.195	0.0012	0.094
Erosion	0.00009	0	0	4.049	10.808	0.0006	0.0198
Histogram Equalization	0.00174	0.00182	0.177	0.0144	0.048	0.0007	0.029

Future Work

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