AIN SHAMS UNIVERSITY

Homomorphic Encryption for Secure Data Computations

A presentation submitted in partial fulfillment of the requirements of
Master of Science in Electrical Engineering
(Computer and Systems Engineering)

By

Mohamed Tarek Ibn Ziad Mohamed Hassan

Supervised By
Dr. Hassan Mohamed Shehata Bedour
Dr. Yousra Mohsen Ali Alkabani

Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Introduction

- Homomorphism comes from the two ancient Greek words; homos (same) and morphe (shape or form).

- Homomorphic encryption (HE) is the kind of encryption, which can be used to perform different arithmetic operations on encrypted data to directly obtain an encrypted result.

- Depending on the number of arithmetic computations that are supported by an algorithm, an HE can be considered as either fully homomorphic encryption (FHE) or partially homomorphic encryption (PHE).

Homomorphic Encryption Importance

- HE is used to build many applications, such as secure voting systems, privacy-preserving face recognition, fingerprint recognition, zero-knowledge watermarking, and location-based services.

- While FHE can help solve privacy issues, it is also desirable to reduce the performance overhead introduced by such methods.

- It is a good practice to utilize PHE techniques in the desired applications, instead of the FHE ones, to avoid such overheads.

Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Thesis Contributions

- **Secure electronic voting (e-voting)**
  - Implementing an e-voting machine, which uses PHE, on a field programmable gate array (FPGA).
  - Injecting a Hardware Trojan (HT) within the FPGA design to tamper voting results.
  - Providing a protection technique against the proposed attack.
  - Showing the different overheads resulting from the protection technique, such as area, timing, and power.

Thesis Contributions (2)

- **Secure FPGA-based designs**
  - Implementing ElGamal encryption scheme and the CRT-based ElGamal (CEG) encryption scheme as a PHE techniques on an FPGA.
  - Showing the resource utilization, timing performance, and power analysis of both schemes.
  - Introducing a dual-circuit design that supports both, multiplicative and additive homomorphic properties and providing the obtained savings on area and power over a regular design that has no resource sharing.

Thesis Contributions (3)

- **Secure image processing**
  - Proposing a secure framework to perform image processing computations over images stored on a third-party server based on Paillier PHE scheme.
  - Supporting image adjustment operations, spatial filtering, edge detection, morphological operations, and histogram equalization.
  - Showing the overheads of the implementation using a Personal Computer (PC) and Mobile device (Mob).

Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Background

- Fully Homomorphic Encryption (FHE)
- Partially Homomorphic Encryption (PHE)
  - ElGamal Scheme
  - CRT-based ElGamal Scheme
  - Paillier Scheme
- Hardware Trojan

Fully Homomorphic Encryption (FHE)

- FHE can perform any operation directly on encrypted data by converting it into a circuit of a certain depth

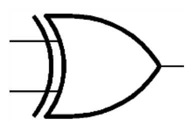
- FHE includes four basic algorithms: `Keygen`, `Encrypt`, `Decrypt`, and `Eval`.

- `Eval` algorithm is built based on three different algorithms: `Add`, `Mult`, and `Recrypt`.

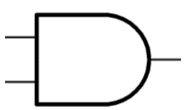
- `Recrypt` operation: cleans the ciphertext from the noise.

Fully Homomorphic Encryption (FHE) (2)

- Why `Add` and `Mult`?
- `XOR` and `AND` is Turing-complete. Any function is a combination of `XOR` and `AND` gates.
- If you can compute sums and products on **encrypted bits**, you can compute any function on encrypted inputs



ADD = XOR



MUL = AND

Fully Homomorphic Encryption (FHE) Drawbacks

- **System complexity**
  - FHE requires a lattice-based cryptosystem that is significantly more complex than PHE cryptosystems.
- **Massive ciphertext sizes**
  - When using recommended security parameters, ciphertexts produced are on the order of 128MB and a public key of 128PB, which are simply not practical.
- **Computation time**
  - The key size is still on the order of several GB, with encryption of a single bit still requiring up to 30 minutes.

- **Solution:** using partially homomorphic encryption (PHE) techniques instead in order to avoid such drawbacks and achieve reasonable outcome.

Partially Homomorphic Encryption (PHE)

- PHE gives the chance to perform only one kind of operations, either addition or multiplication, on ciphertexts without revealing data.

$$E(m_1)\ \boldsymbol{Op}\ E(m_2) = E(m_1 \times m_2)$$ **Multiplicative homomorphism**

$$E(m_1)\ \boldsymbol{Op}\ E(m_2) = E(m_1 + m_2)$$ **Additive homomorphism**

ElGamal Scheme

- Key generation:
  - The secret key (k)
  - The public key (g, h), $where\ h = g^k \ mod\ n$
    - k and g are random numbers. n is a large prime.
- Encryption:
  - $C_1 = g^l \ (mod\ n)$ and $C_2 = h^l \times m\ (mod\ n)$
    - l is a random number.
- Decryption:
  - $m = C_1^{-k} \times C_2 \ (mod\ n)$
- It is a **multiplicative homomorphic** scheme.
  - If (x_1, y_1) and (x_2, y_2) are valid encryptions for m_1 and m_2, with the same key, then $(x_1\ x_2,\ y_1\ y_2)$ is a valid encryption of $m_1\ m_2$.

ElGamal Scheme

- Key generation:
  - The secret key (k)
  - The public key (g, h), $where\ h = g^k \ mod\ n$
    - k and g are random numbers. n is a large prime.
- Encryption:
  - $C_1 = g^l \ (mod\ n)$ and $C_2 = h^l \times m\ (mod\ n)$
    - l is a random number.
- Decryption:
  - $m = C_1^{-k} \times C_2 \ (mod\ n)$

$$\because (x_1, y_1) = \left(g^l, h^l \times m_1\right)$$
$$\because (x_2, y_2) = \left(g^{l'}, h^{l'} \times m_2\right)$$
$$\therefore (x_1 x_2, y_1 y_2) = (g^{l+l'}, h^{l+l'} \times m_1 m_2)$$

CRT-based ElGamal (CEG) Scheme

- Key generation:
  - The secret key (k)
  - The public key (g, h), $where\ h = g^k \ mod\ n$
- Encryption:
  - $C_1 = g^{l_i} \ (mod\ n)$ and $C_2 = h^{l_i} \times g^{m_i}\ (mod\ n)$
    - where $m_i = m\ (mod\ d_i\ ),\ d_i\ \ is\ a\ random\ number,\ i = 1, \ldots, t$ and $\gcd(d_i, d_j) = 1\ for\ i\ \neq j$
- Decryption:
  - $m = CRT^{-1}\left[\left(\log_g\left(C_{2_i} \times C_{1_i}^{-k}\ \ (mod\ n)\right), i = 1, \ldots, t\right)\right]$
    - $CRT^{-1}\left[C_i\right] = \sum_{i=1}^{t} C_i \frac{d}{d_i}\left(\frac{d}{d_i}^{-1} mod\ d_i\right) mod\ d$
- It is an **additive homomorphic** scheme that uses the Chinese Remainder Theorem (CRT).

CRT-based ElGamal (CEG) Scheme

- Key generation:
  - The secret key (k)
  - The public key (g, h), $where\ h = g^k \ mod\ n$
- Encryption:
  - $C_1 = g^{l_i} \ (mod\ n)$ and $C_2 = h^{l_i} \times g^{m_i}\ (mod\ n)$
    - where $m_i = m\ (mod\ d_i\ ),\ d_i\ \ is\ a\ random\ number,\ i = 1, \dots, t$ and $\gcd(d_i, d_j) = 1\ for\ i\ \neq j$
- Decryption:
  - $m = CRT^{-1}\left[\left(\log_g\left(C_{2_i} \times C_{1_i}^{-k}\ (mod\ n)\right), i = 1, \dots, t\right)\right]$
    - $CRT^{-1}\,[C_i] = \sum_{i=1}^{t} C_i \frac{d}{d_i}\left(\frac{d}{d_i}^{-1} mod\ d_i\right) mod\ d$

$$\because (x_1, y_1) = \left(g^{l}, h^{l} \times g^{m_1}\right)$$
$$\because (x_2, y_2) = \left(g^{l'}, h^{l'} \times g^{m_2}\right)$$
$$\therefore (x_1\, x_2, y_1 y_2) = (g^{l+l'}, h^{l+l'} \times g^{m_1+m_2})$$

Paillier Scheme

- Key generation:
  - The secret key (λ), $where\ \lambda = lcm(p-1, q-1)$
  - The public key (g, N)
    - $where\ h = gcd(L(g\lambda\ mod\ N^2), N) = 1$ and $L(u) = \frac{u-1}{N}$
- Encryption:
  - $C = g^m\ r^N\ (mod\ N^2)$
- Decryption:
  - $m = \frac{L(C^\lambda\ mod\ N^2)}{L(g^\lambda\ mod\ N^2)}\ mod\ N$
- It is an **additive homomorphic** scheme. It also supports a self-blinding operation, which allows multiplication of encrypted integer by a plaintext scalar.

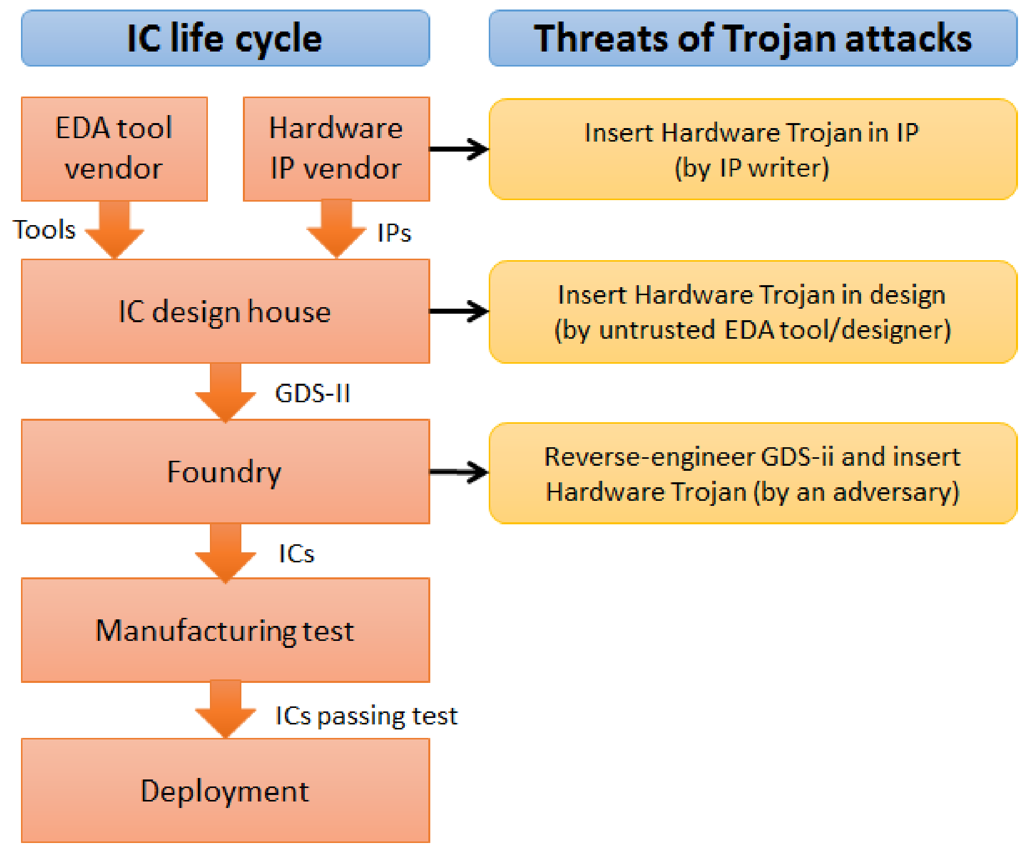
Paillier Scheme

- Key generation:
  - The secret key (λ), $where\ \lambda = lcm(p-1, q-1)$
  - The public key (g, N)
    - $where\ h = gcd(L(g\lambda\ mod\ N^2), N) = 1$ and $L(u) = \frac{u-1}{N}$
- Encryption:
  - $C = g^m\ r^N\ (mod\ N^2)$
- Decryption:
  - $m = \frac{L(C^\lambda\ mod\ N^2)}{L(g^\lambda\ mod\ N^2)}\ mod\ N$

$$\because x_1 = g^{m_1}\ r^N$$
$$\because x_2 = g^{m_2}\ r'^N$$
$$\therefore x_1 x_2 = g^{m_1+m_2}\ r^N r'^N$$

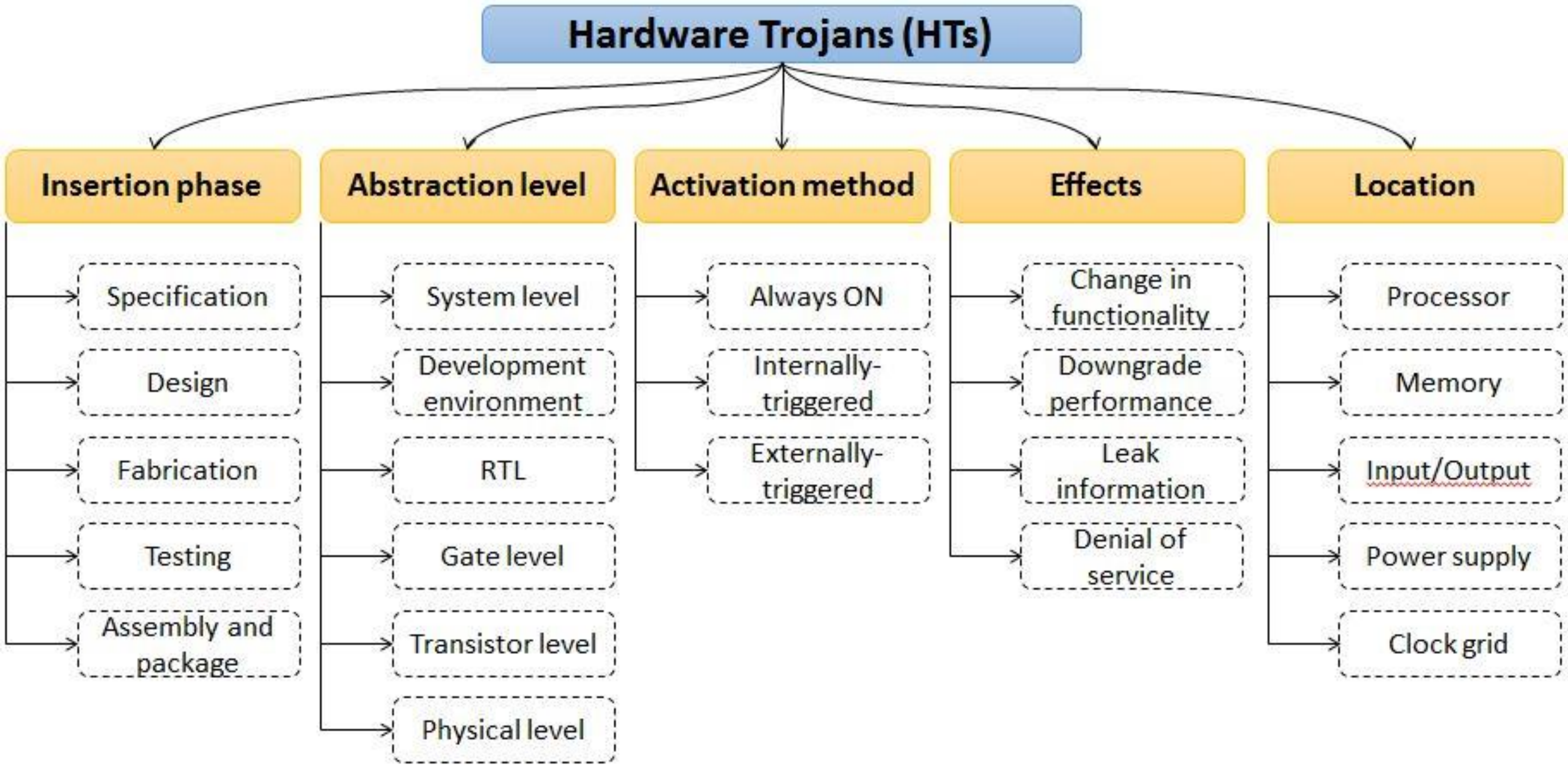
Hardware Trojan

- Hardware Trojan is a malicious alteration of one's own hardware. This alternation may, under specific rare circumstances, result in information leakage out of the system or functional changes of the system itself



Hardware Trojan (2)

- Hardware Trojan Taxonomy.



Outline

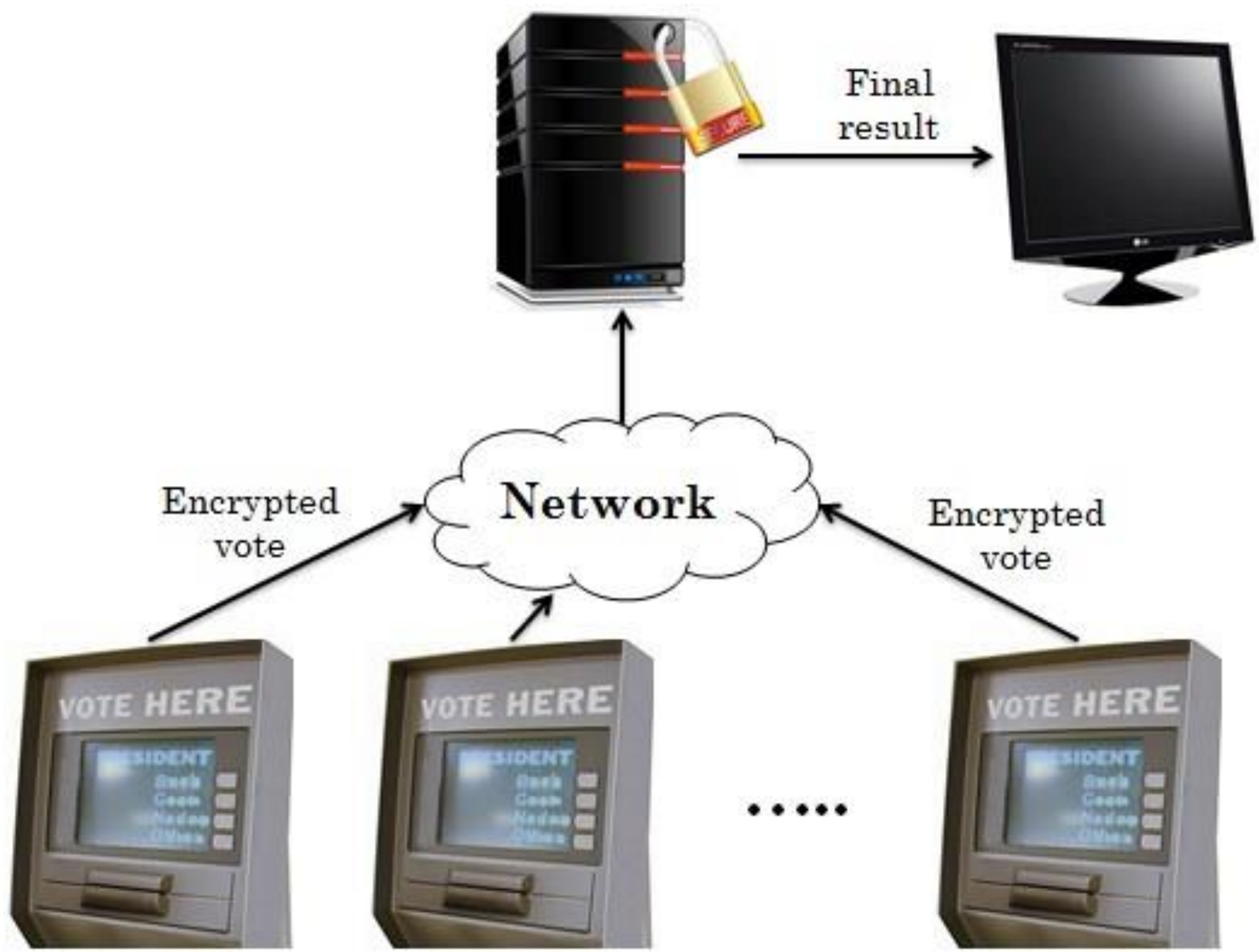
- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

E-voting Attacks and Countermeasures

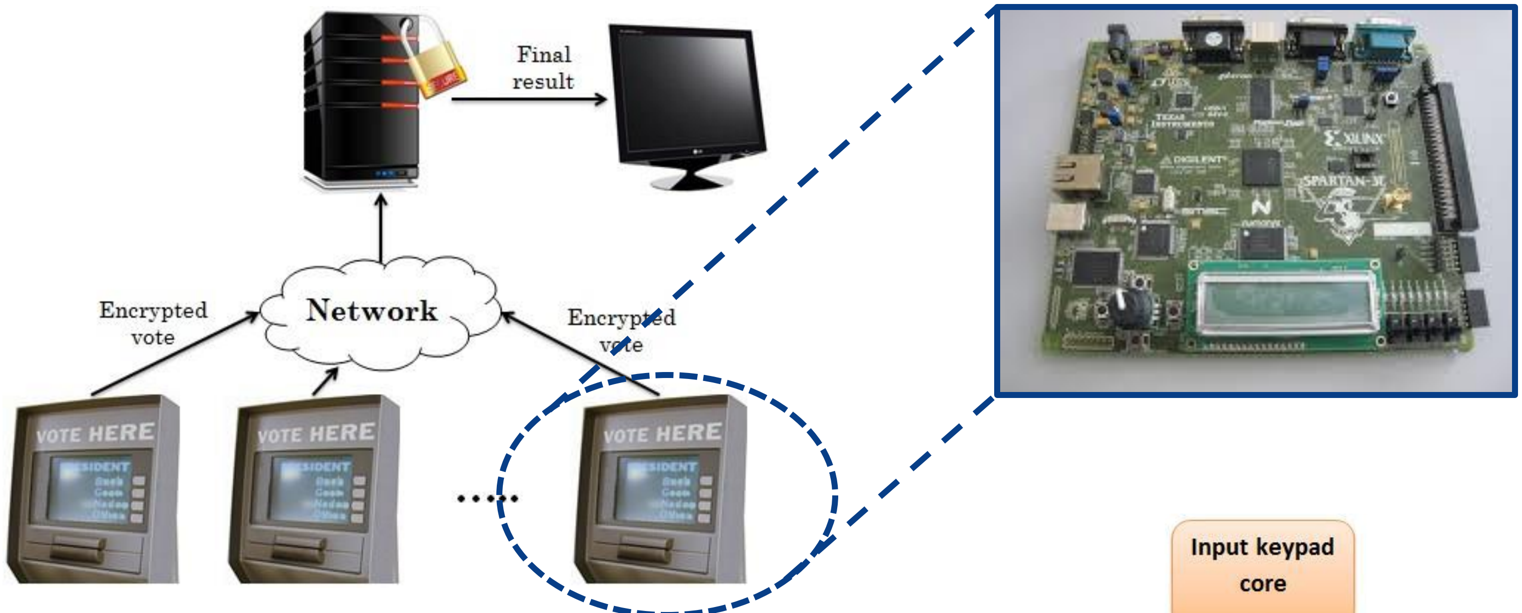
- E-voting systems have started to be widely used as they do offer various advantages over the traditional voting methods.

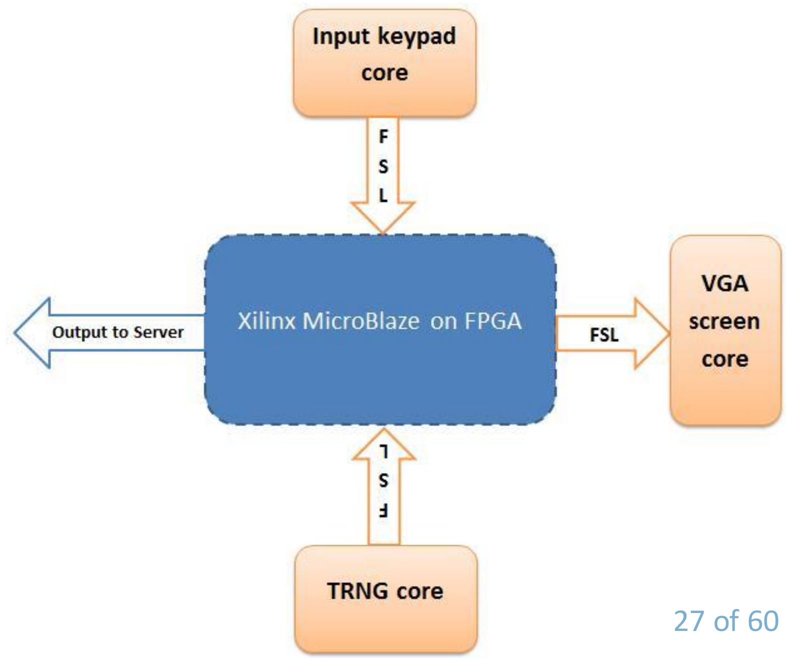
- However, e-voting also introduces many security challenges that need to be handled wisely, otherwise, it might bomb the whole voting process.
- E-voting machines may contain harmful back-doors, which can affect the dependability of the system.

E-voting System Overview



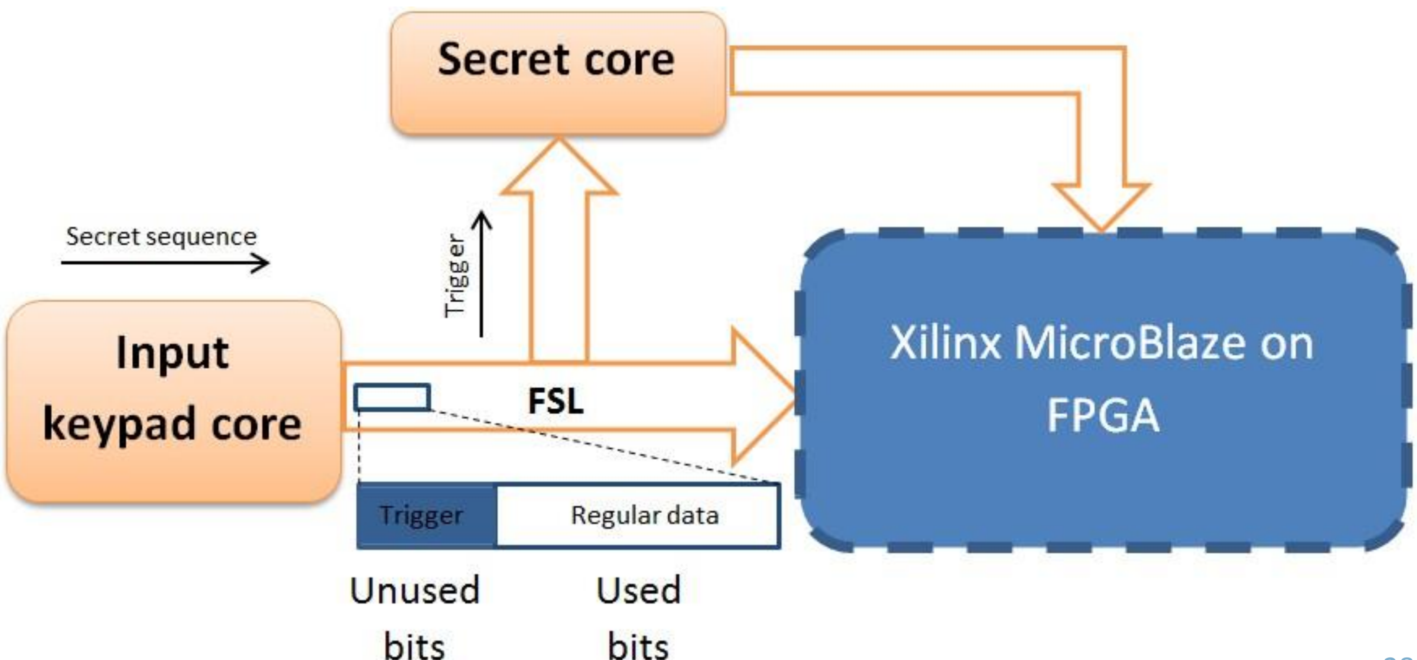
E-voting System Overview (2)





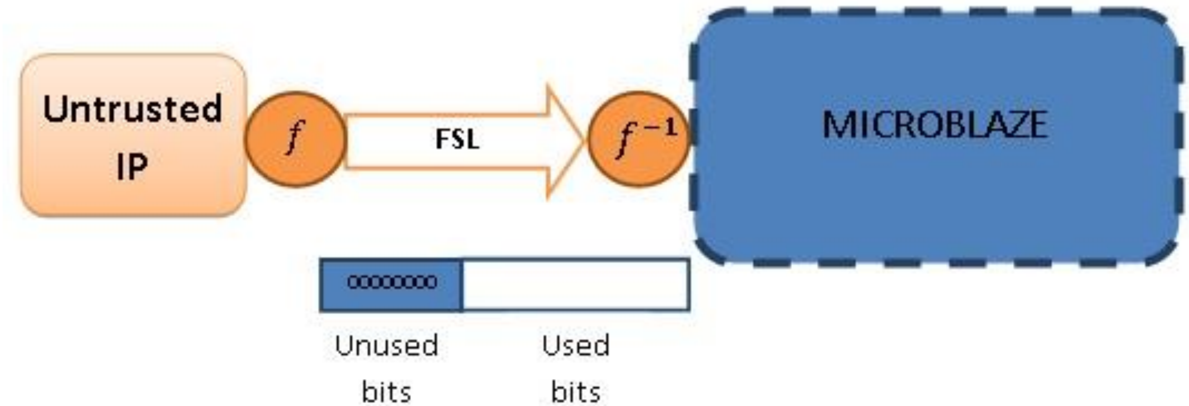
Scenario for a Possible Attack

- An untrusted FPGA-based voting machine may be used to tamper with the legal votes of users.
- The attacker may add a hidden core, connected to the MicroBlaze core, that replaces the user's vote with another one, if it receives a special external trigger.



Protection Against Proposed Attack

- **First solution**: Resetting unused bits
  - We reset any unused bits to zero before receiving them at the MicroBlaze



- **Second solution**: The enhanced Simple Blockage (SB) method.
  - Here, we choose to protect the design using a simple xoring function.
  - Obfuscation will take place between keypad and MicroBlaze.

Evaluation (Resetting Unused Bits)

- Device utilization for untrusted and protected systems (with and without resetting unused bits) showing overhead percentage on a Xilinx XC3S500E Spartan-3E FPGA.

Logic resources	Untrusted system Without resetting unused bits	Protected system With resetting unused bits	Overhead (%)
No. of used slice flip flops	3,401	3,428	0.79
No. of used 4-input LUTs	4,266	4,396	3.05
Total no. of used 4-input LUTs	4,391	4,521	2.96

- From the timing perspective, the untrusted design achieves max frequency of 59.677 MHz, while the protected design achieves a max frequency of 63.107 MHz.
- So, delay overhead is 0.206 ns, which is below 10%.

Evaluation (Resetting Unused Bits) (2)

- Power comparison between original and protected systems (with and without resetting unused bits) on a Xilinx XC3S500E Spartan-3E FPGA using Xilinx Power Analyzer.

Logic resources	Power consumption (W)	
	Untrusted system Without resetting unused bits	Protected system With resetting unused bits
Logic	0.009	0.009
Signals	0.007	0.008
BRAMs	0.006	0.006
MULTs	0.001	0.001
DCMs	0.041	0.043
IOs	0.340	0.340
Leakage	0.094	0.094
Total	0.498	0.501

Evaluation (Enhanced SB Method)

- Device utilization for untrusted and protected systems (with and without enhanced Simple Blockage) showing overhead percentage on a Xilinx XC3S500E Spartan-3E FPGA.

Logic resources	Untrusted system Without enhanced SB	Protected system With enhanced SB	Overhead (%)
No. of used slice flip flops	3,401	3,436	1.03
No. of used 4-input LUTs	4,266	4,437	4.00
Total no. of used 4-input LUTs	4,391	4,562	3.89

- From the timing perspective, the untrusted design achieves max frequency of 59.677 MHz, while the protected design achieves a max frequency of 50.666 MHz.
- So, delay overhead is 0.205 ns.

Evaluation (Enhanced SB Method) (2)

- Power comparison between original and protected systems (with and without enhanced Simple Blockage) on a Xilinx XC3S500E Spartan-3E FPGA using Xilinx Power Analyzer.

Logic resources	Power consumption (W)	
	Untrusted system Without enhanced SB	Protected system With enhanced SB
Logic	0.009	0.009
Signals	0.007	0.007
BRAMs	0.006	0.006
MULTs	0.001	0.001
DCMs	0.041	0.043
IOs	0.340	0.340
Leakage	0.094	0.094
Total	0.498	0.500

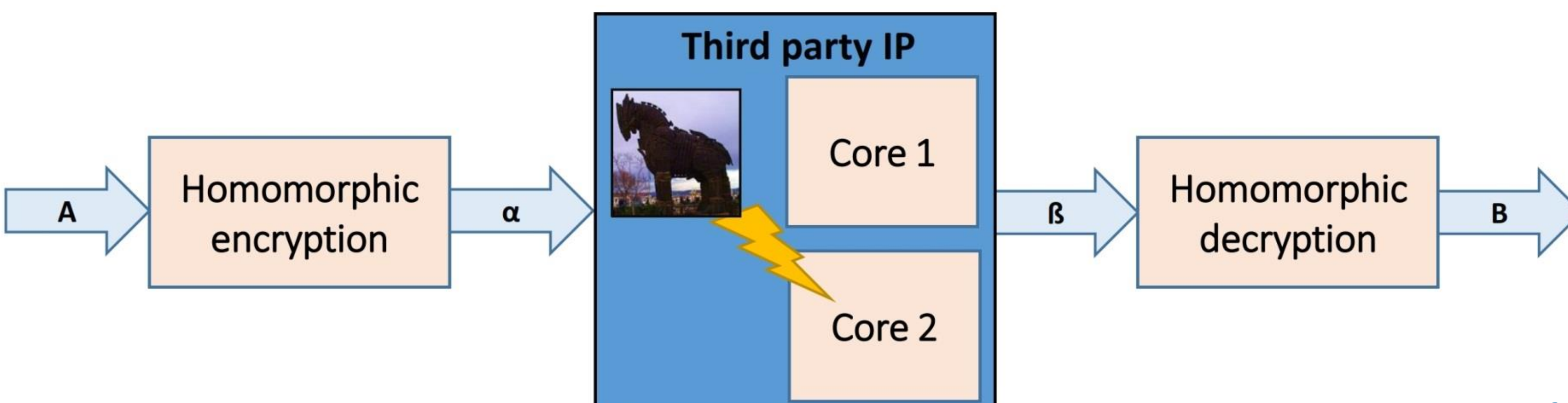
Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Protection against Hardware Trojans

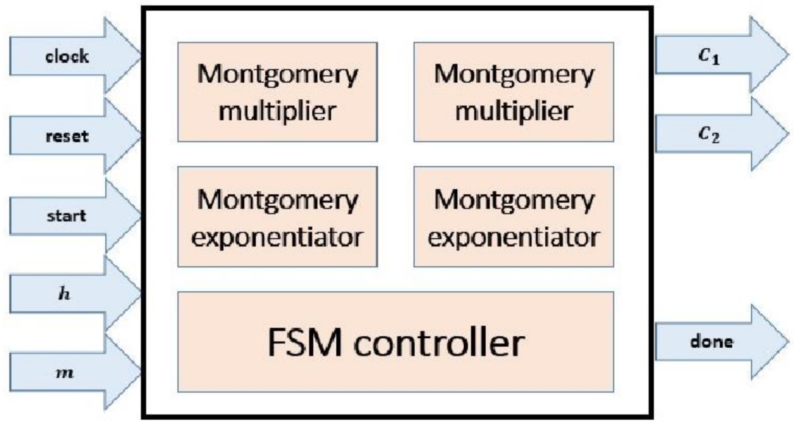
- Maintaining technology secrets of the fabrication facilities and design royalties of third party IP owners raises the difficulty of Hardware Trojan detection and protection.

- Homomorphic encryption may be used to solve this issue and defeat Hardware Trojans.

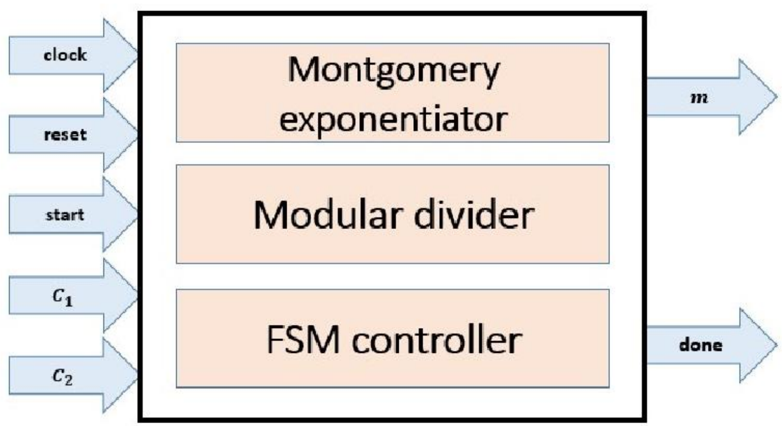


HT Protection using PHE Methods (ElGamal Scheme)

- The block diagram of our implementation of ElGamal encryption/decryption scheme.



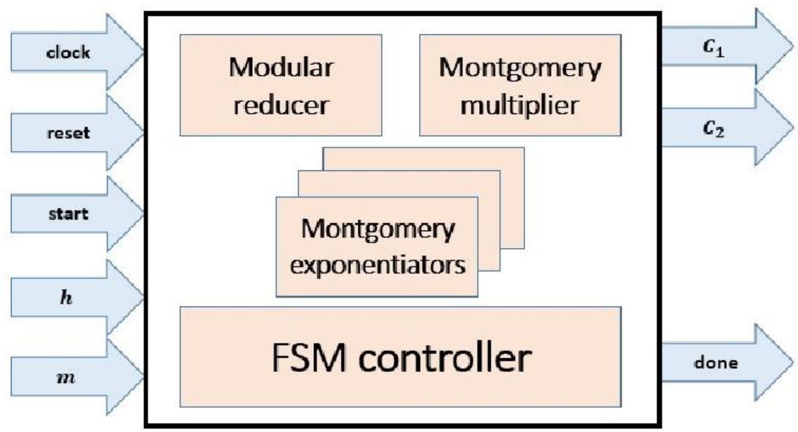
(A) Encryption.



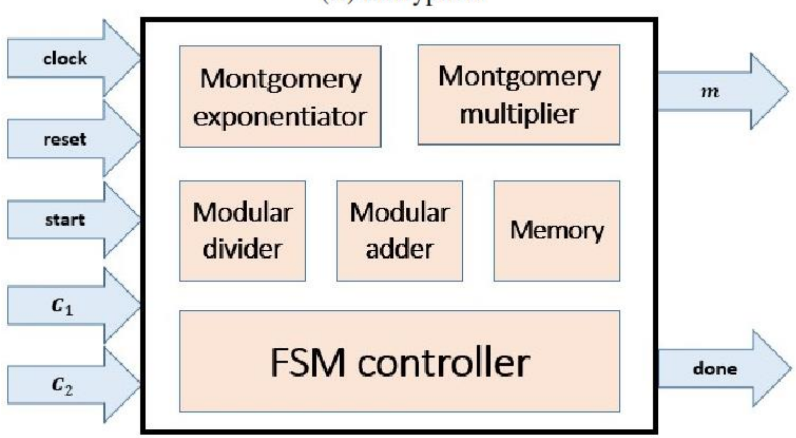
(B) Decryption.

HT Protection using PHE Methods (CEG Scheme)

- The block diagram of our implementation of CEG encryption/decryption scheme.



(A) Encryption.



(B) Decryption.

HT Protection using PHE Methods (Dual-Circuit Design)

- Some third party IPs require the usage of more than one single type of operation. Ex: an ALU that uses a selection line to switch its mode between two different operations.

- We suggest a solution by combining the two previously schemes, ElGamal and the CEG, in a single dual-circuit design. Thus, the proposed design supports both additive and multiplicative homomorphism.

- We try to share resources as much as we can between the two schemes in order to have minimal design cost.

Evaluation (PHE Methods)

- Resource utilization of ElGamal and CEG encryption/decryption schemes for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

Logic resources	Encryption		Decryption	
	ElGamal	CEG	ElGamal	CEG
Number of Registers	295	614	207	364
Number of LUTs	420	715	259	442
Number of BRAMs	0	0	0	1

- Timing performance of ElGamal and CEG encryption/decryption schemes for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

	Encryption		Decryption	
	ElGamal	CEG	ElGamal	CEG
Frequency (MHz)	161.277	164.352	123.870	121.862
No. of cycles	171	480	153	512

Evaluation (PHE Methods) (2)

- Power consumption (mW) of ElGamal and CEG encryption/decryption schemes for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

Logic resources	Encryption		Decryption	
	ElGamal	CEG	ElGamal	CEG
Logic	3.84	5.47	2.70	3.69
Signals	2.82	4.69	2.01	3.23
BRAMs	0.00	0.00	0.00	0.74
IOs	16.51	8.99	5.23	2.74
Clocks	5.65	7.87	4.21	5.87
Leakage	65.00	65.00	64.00	64.00
Total	93.82	92.02	78.15	80.27

Evaluation (Dual-Circuit Design)

- Area reduction of our dual ElGamal design over the regular ElGamal design for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

Logic resources	Encryption			Decryption		
	Regular ElGamal	Dual ElGamal	Area reduction (%)	Regular ElGamal	Dual ElGamal	Area reduction (%)
Registers	909	635	30.14	536	364	32.09
LUTs	1137	735	35.36	626	457	26.99
BRAMs	0	0	00.00	1	1	00.00

$$Reduction\ space(\%) = \frac{Regular\ area - Dual\ area}{Regular\ area} \times 100.$$

Evaluation (Dual-Circuit Design) (2)

- Timing comparisons between our dual ElGamal design and the regular ElGamal design for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

	Encryption		Decryption	
	Regular ElGamal	Dual ElGamal	Regular ElGamal	Dual ElGamal
Frequency (MHz)	161.277	158.51	117.099	121.344
No. of cycles	651	662	665	665

Evaluation (Dual-Circuit Design) (3)

- Power consumption (mW) of our dual ElGamal design and the regular ElGamal design for k = 8 bits on Xilinx Spartan-6 XC6SLX75 FPGA.

Logic resources	Encryption		Decryption	
	Regular ElGamal	Dual ElGamal	Regular ElGamal	Dual ElGamal
Logic	9.25	6.29	5.91	3.82
Signals	8.14	6.02	5.67	3.49
BRAMs	0.00	0.00	0.74	0.74
IOs	25.27	10.83	5.67	3.61
Clocks	11.78	6.89	8.78	4.86
Leakage	65.00	65.00	65.00	64.00
Total	119.44	95.03	91.77	80.52

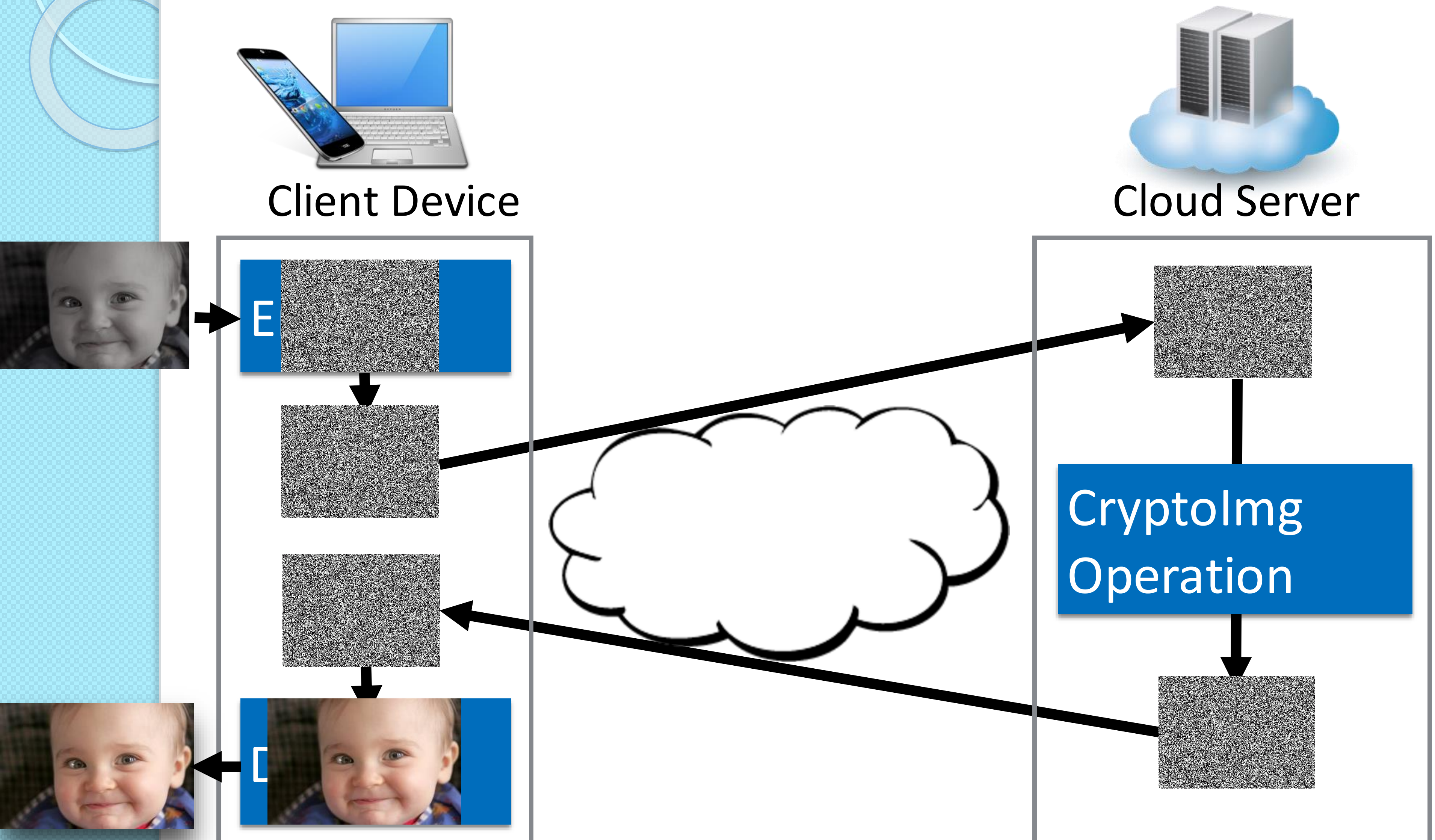
Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Processing over Encrypted Images

- 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 data. Outsourcing the raw data exposes a lot of information.
- How to protect user's privacy while editing images in the cloud?

CryptoImg System Overview



CryptoImg Operations

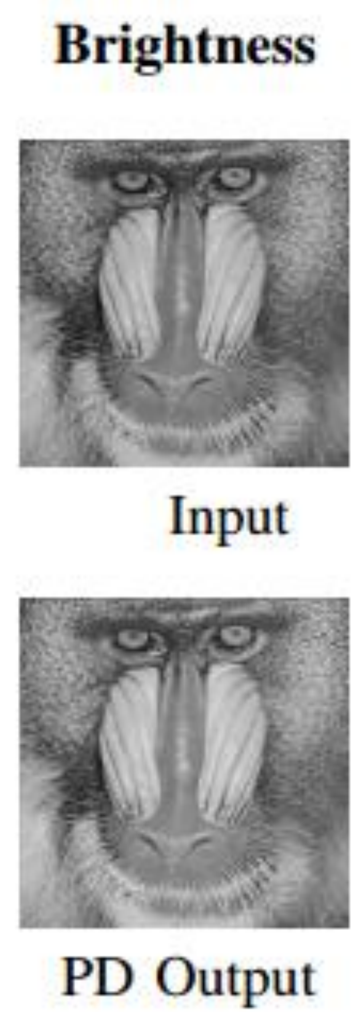
- **CyrptoImg** supports the following image processing operations:
  - Image Adjustment
  - Noise Reduction
  - Edge Detection
  - Morphological Operations
  - Histogram Equalization
- **Problem**: Paillier scheme is defined over the group of positive integers. In practice, we also need to deal with negative and real numbers.
- **Solution**: Use an encoding scheme that maps negative and real numbers to integers and preserves the Paillier encryption homomorphic properties.

Secure Image Adjustment

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

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

- Client decrypts the result.



Secure Noise Reduction

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

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

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

- Client decrypts the result.

LPF

Input

PD Output

Secure Edge Detection

- Client encrypts the source image `I`.

- Servers computes the encrypted horizontal and vertical gradients of image.

$$\llbracket G_x(u,v) \rrbracket = \sum_{u=1,v=1}^{m,n} h_1(u,v) \otimes \llbracket I(u,v) \rrbracket$$

$$\llbracket G_y(u,v) \rrbracket = \sum_{u=1,v=1}^{m,n} h_2(u,v) \otimes \llbracket I(u,v) \rrbracket$$

- Client decrypts the result to compute the gradient magnitude and direction.

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

$$\Theta = \mathrm{atan2}\left(G_y, G_x\right)$$

Input

PD Output

Secure Morphological Operations

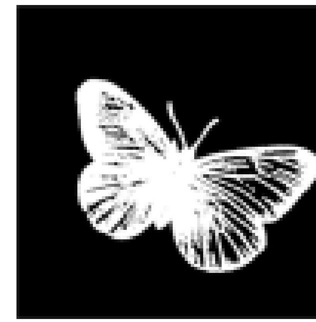
- Client encrypts the source image I.

- Servers computes, and sends it to client.

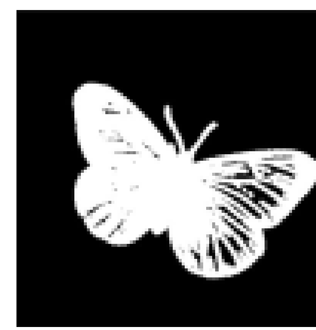
$$[\![L(u,v)]\!] = \sum_{u=1,v=1}^{m,n} [\![I(u,v)]\!]$$

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

Dilation



Input



PD Output

Secure Histogram Equalization

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

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

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

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

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

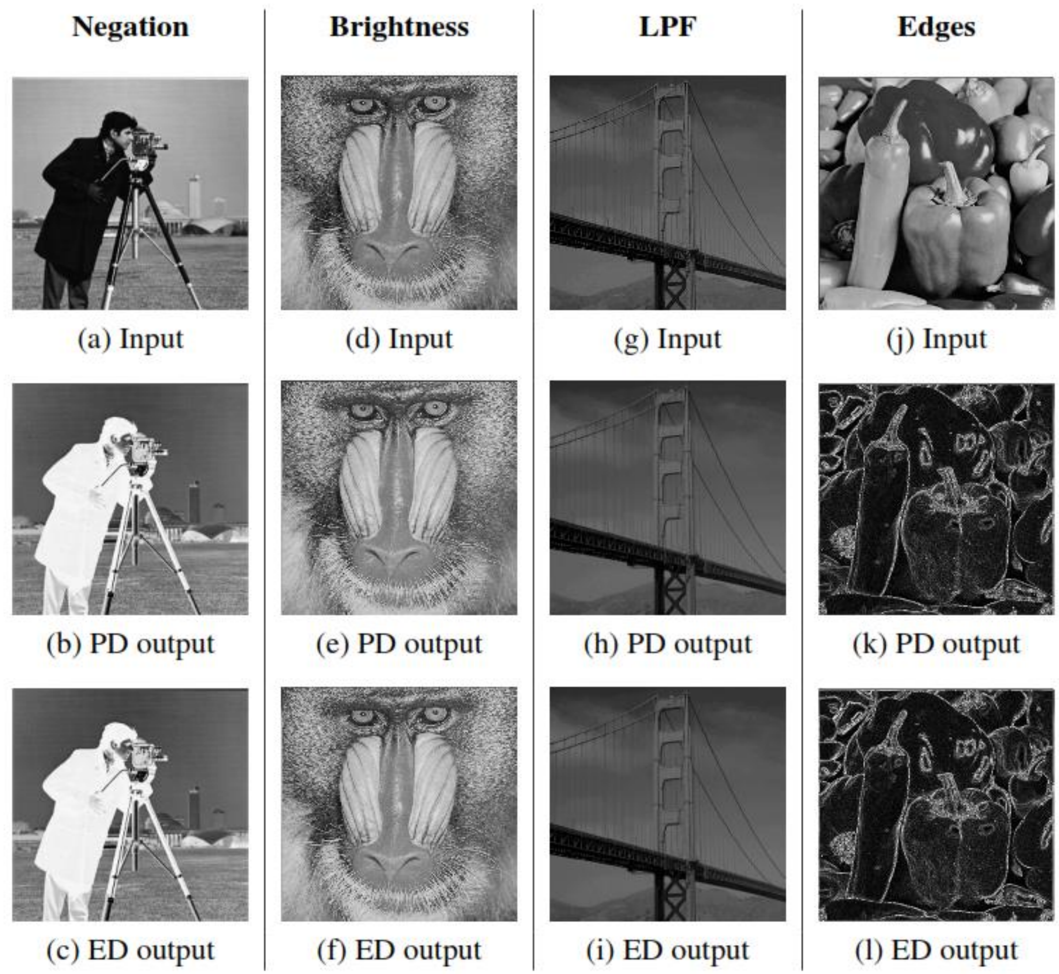
Equalization

Input

PD Output

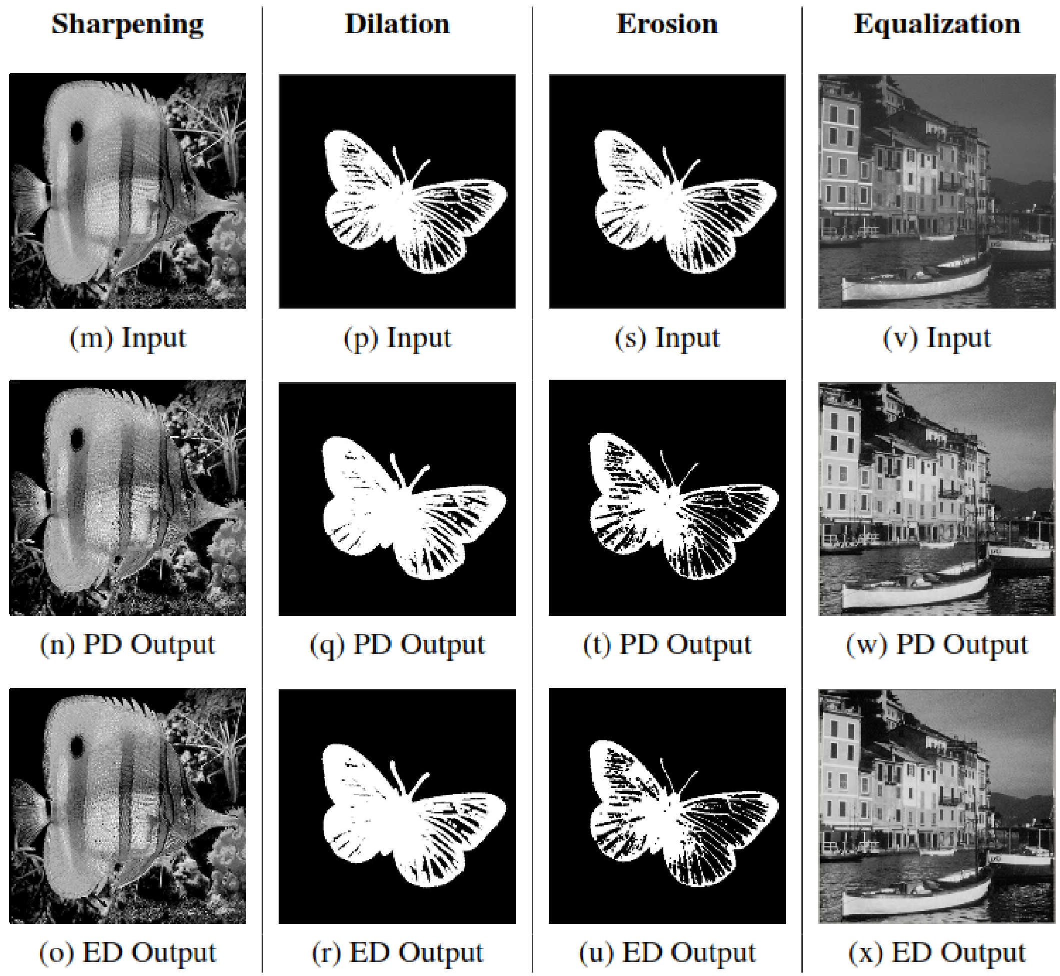
Evaluation (Visual Output Results)

- CryptoImg is implemented as an extension for `OpenCV` library.



Evaluation (Visual Output Results) (2)

- CryptoImg is implemented as an extension for `OpenCV` library.



Evaluation (Computation Time Results)

- Execution Time (sec) of the Paillier encryption/decryption of image using different key sizes on both personal computer (PC) and mobile device (Mob) clients. We used 512×512 images for PC and 256×256 images for Mob.

Key Size	256	512	1024	2048
Encrypt-PC	23.9164	156.905	1154.29	7670.49
Decrypt-PC	1.39223	1.93554	4.06813	9.62313
Encrypt-Mob	13	73	575	3701
Decrypt-Mob	10	48	325	2268

Evaluation (Computation Time Results) (2)

- Execution Time (sec) of the proposed operations using 1024-bit and 2048-bit keys on both personal computer (PC) and mobile device (Mob) clients in plaintext domain (PD) and encrypted domain (ED). The server is modeled as the PC. We used 512 $\times$ 512 images.

Operation	PD	ED					
		Pre-processing		Server		Post-processing	
		PC	Mob	1024-bit	2048-bit	PC	Mob
Negation	0.00122	0	0	42.4737	137.925	0	0
Brightness	0.00108	0	0	0.81994	2.39777	0	0
LPF	0.00763	0	0	180.508	609.199	0	0
Sobel filter	0.00642	0	0	147.567	482.195	0.0012	0.0940
Sharpening	0.00977	0	0	238.257	807.528	0	0
Dilation	0.00008	0	0	4.04937	10.8085	0.0005	0.0198
Erosion	0.00009	0	0	4.04937	10.8085	0.0006	0.0198
Equalization	0.00174	0.00182	0.177	0.01446	0.04835	0.0007	0.0290

Outline

- Introduction
- Thesis Contributions
- Background
- E-voting Attacks and Countermeasures
- Protection against Hardware Trojans
- Processing over Encrypted Images
- Conclusion

Conclusion

- We tackled the problem of computing securely over encrypted data.
- Instead of going through the non-practical techniques of FHE, our target was to implement PHE methods and extend their functionality.
- We applied our idea on three different cases.
  - Securing e-voting machines against intruders.
  - Securing FPGA-based designs against untrusted third party IPs .
  - Securing image processing operations over untrusted clouds.
- The overheads accompanied by using such techniques are reasonable compared to the huge overheads of the FHE techniques reported in the literature.

Publications

- **M. Tarek Ibn Ziad**, A. Al-Anwar, **Y. Alkabani**, M. W. El-Kharashi, and **H. Bedour**. "E-voting attacks and countermeasures". In *Proceedings of the 10th International Symposium on Frontiers of Information Systems and Network Aplications (FINA 2014), held in conjunction with the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA-2014)*, pages 269–274, Victoria, BC, Canada, May 13–16, 2014.

- **M. Tarek Ibn Ziad**, A. Alanwar, **Y. Alkabani**, M. W. El-Kharashi, and **H. Bedour**. "Homomorphic data isolation for hardware Trojan protection". In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 131–136, Montpellier, France, July 8–10, 2015.

Thank You