Homomorphic Data Isolation for Hardware Trojan Protection

M. Tarek Ibn Ziad*, Amr Alanwar†, Yousra Alkabani*, M. Watheq El-Kharashi* and Hassan Bedour*

*Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt

†Department of Electrical Engineering, UCLA, Los Angeles, CA, USA

Email: mohamed.tarek@eng.asu.edu.eg, alanwar@ucla.edu,

{yousra.alkabani, watheq.elkharashi, hassan.bedour}@eng.asu.edu.eg

Abstract—The interest in homomorphic encryption/decryption is increasing due to its excellent security properties and operating facilities. It allows operating on data without revealing its content. In this work, we suggest using homomorphism for Hardware Trojan protection. We implement two partial homomorphic designs based on ElGamal encryption/decryption scheme. The first design is a multiplicative homomorphic, whereas the second one is an additive homomorphic. We implement the proposed designs on a low-cost Xilinx Spartan-6 FPGA. Area utilization, delay, and power consumption are reported for both designs. Furthermore, we introduce a dual-circuit design that combines the two earlier designs using resource sharing in order to have minimum area cost. Experimental results show that our dual-circuit design saves 35% of the logic resources compared to a regular design without resource sharing. The saving in power consumption is 20%, whereas the number of cycles needed remains almost the same.

Keywords- ElGamal Encryption, Hardware Trojan, Homomorphism, Security

I. INTRODUCTION

Increasing the complexity of systems proclaims the outsourced manufacturing concept nowadays. This raises a lot of trust issues in the design industry in many directions. Anyone with access to any step of the manufacturing process could alter the final product to inject a Hardware Trojan. Malicious circuitry can be injected by fabrication facilities or third party IP owners. This threatens design community as the fabrication process is obscured from designers and the details of third party IPs are hidden to protect IP owners' rights [1].

Hardware Trojan appears to be one of the most important topics as the use of silicon chips in different applications becomes very popular, varying from cell phones, cars, to strategically important military devices. It is important to provide methods that resolve the trust issues between fabrication facilities, designers, and end-users. End-users need to make sure that products are not controlled by unknown entities, are stable enough, and will not leak critical information. 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.

In general, homomorphic encryption is a type of encryption, which allows specific types of operations to be carried out

on ciphertext and generates an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This is a desirable feature that has been utilized in many modern systems [2], [3]. In this work, we introduce the idea of using homomorphism to defeat Hardware Trojan injected in third party IPs. Consider the case where a third party IP is needed to carry out some operation on data A and will produce output data B. Fig. 1 shows the ideal world, where the third party IP does not have any access to the real data as it is homomorphically encrypted. This will give us the capability to carry out the required operation by the third party IP without revealing the original data. Thus, we can retrieve the result B after the decryption process. Full homomorphism (FH) evaluates any arbitrary depth circuit on ciphertexts, whereas partial homomorphism (PH) supports one type of operations only, addition or multiplication.

Fig. 1: Homomorphic encryption to protect from hardware Trojan.

The key contributions of this paper include:
1) Discussing new ideas to have a blind data processing by the third party IP with a minimum cost.
2) Implementing ElGamal encryption scheme, which is multiplicative homomorphic and the CRT-based Elgamal (CEG) encryption scheme, which is additive homomorphic, on a low-cost FPGA and showing the resource utilization, performance, and power analysis of both schemes.
3) 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.

The rest of the paper is organized as follows. Section II summarizes the related work. Overview about homomorphism and the utilized schemes are given in Section III. Hardware Trojan protection using PH is introduced in Section IV. Experimental evaluation and estimation of the overhead of the proposed methods are shown in Section V. Section VI concludes the work.

IEEE computer society

II. RELATED WORK

Recently, some architectural methodologies try to increase the chances of the activation of a Hardware Trojan during testing. Salmani *et. al* increased the Trojan activity by inserting dummy flip-flops in the design [4]. They chose the locations of the inserted flip-flops based on a transition probability threshold. Rajendran *et al.* introduced a methodology for securing all the gates of the design using ring oscillators [5]. They added extra logic that converts paths of the circuit into ring oscillators. Changes in the frequency of the ring oscillators were used to detect the presence of Trojans. Al-Anwar *et al.* in [6] developed a novel method for the protection against a hardware Spyware that depends basically on decreasing the probability of seeking sensitive information. They introduced multiplexing between multiple variants implementation. Then, they used cyclic redundancy check (CRC) to detect the infected IP. In [7], Al-Anwar *et al.* suggested obfuscating the output of the suspected IP before sending out data, then undoing that obfuscation at the input of the receiver in order to protect data from leaking and avoid injected triggering. They introduced using either RC4 or a simple obfuscating function. Ibn Ziad *et al.* injects a Hardware Trojan in a voting machine to tamper voting results [8]. The attack depends mainly on the unused bits. They provided a protection technique against the proposed attack and showed its overhead. Al-Anwar *et al.* in [9] introduced multiplexing reconfigurable IPs' outputs and CRC Trojan detection scheme (MCRC) method in order to decrease the probability of leaking critical information by a hacked IP. They also suggested using partial reconfiguration technology to remove an infected IP.

Side-channel dependent methodologies for Hardware Trojan detection aim to localize the impact of the Trojan on the circuit without activating it. Their main idea is to try to detect the presence of a Trojan with high probability via detecting the overload of the Trojan circuit on different circuit parameters; such as the delay or the power as compared to a non-infected circuit. Rad *et al.* studied the impact of a Trojan on the power supply transient current of an IC using statistical methods [10]. Jin and Makris used path delay analysis to detect Trojans [11]. Moreover, gate-level characterization techniques accompanied by statistical methods were used to detect Hardware Trojans [12].

Unfortunately, all Hardware Trojan detection methods require the presence of a non-infected (golden) chip. That requirement represents a real problem as it is feasible only if the design does not contain third party IPs [13]. But, if the system designer integrates third party IPs in the design, these methods become less practical. Zhang and Tehranipoor tried to provide an alternative to using a golden design by using code coverage analysis, formal verification, and ATPG methods to achieve high confidence in whether the circuit is Trojan-free or Trojan-inserted [14]. Baumgarten *et al.* suggested using reconfigurable logic barriers within a design to prevent the activation and operation of Hardware Trojans inserted during the manufacturing stage of an IC [15].

III. BACKGROUND

The aim of this section is to give a brief description about the idea of homomorphism, survey existing partial homomorphic encryption schemes, and discuss ElGamal security scheme.

A. Partial Homomorphism (PH)

PH has been known for many years. It offers the ability to perform a certain type of operations, addition or multiplication, on ciphertexts without revealing data. For example, let us consider the two messages, m_1 and m_2, where both messages are encrypted and their ciphertexts are given by $E(m_1)$ and $E(m_2)$, respectively. If the multiplication of the two ciphertexts is equivalent to the ciphertext of the multiplication of the two messages as shown in (1), we call this a multiplicative homomorphic scheme. On the other hand, if the multiplication of the two ciphertexts equals the ciphertext of the addition of the two messages as shown in (2), we call this an additive homomorphic scheme.

$$E(m_1) \times E(m_2) = E(m_1 \times m_2) \tag{1}$$

$$E(m_1) \times E(m_2) = E(m_1 + m_2) \tag{2}$$

It is worth mentioning that PH is different from FH, which allows the efficient evaluation of an arbitrary depth circuit (composed of additions and multiplications) to be evaluated directly on ciphertexts. The first full homomorphic encryption (FHE) scheme was introduced by Gentry [16] in 2009. Since then, there has been some work done toward obtaining efficient hardware implementations of FHE schemes. Hardware building blocks for the lattice-based cryptosystem were considered by Göttert *et al.* [17]. Also, Pöppelmann and Güneysu introduced an efficient hardware implementation of ring-learning-with-errors (RLWE) based encryption [18]. However, these schemes are not practical for this application due to its very large ciphertext and public key sizes. Thus, we focus on PH in this paper.

One of the earliest discoveries in the context of PH is the Goldwasser-Micali cryptosystem [19], whose security is based on the quadratic residuosity problem. It allows homomorphic evaluation of a bitwise exclusive-or. Other additive homomorphic encryption schemes that provide semantic security are Benaloh [20] and Paillier [21]. On the other hand, there exist two well-known schemes, which are multiplicative homomorphic schemes. The first one is the Rivest-Shamir-Adleman (RSA) [22], which is one of the most widely used public-key cryptosystems. The second is ElGamal encryption scheme [23], which is the selected cryptosystem to be used in our work.

B. ElGamal Scheme

ElGamal public-key cryptography algorithm is considered to be one of the efficient and popular algorithms that provides a high level of security. To illustrate its functionality, let us consider that a user called *Alice* wants to send a private

message m to another user *Bob*. ElGamal process works as follows. *Bob* generates his keys. He chooses a secret random exponent k and a generator g. So, his public key is (g, h) where $h = g^k (mod\, n)$ and n is a large prime. *Alice* has to encrypt the message m before sending it to *Bob*. She generates a random exponent l and sends the ordered pair (C_1, C_2) to *Bob*, where C_1 and C_2 are defined as (3).

$$C_1 = g^l(mod\, n), C_2 = h^l \times m(mod\, n) \qquad (3)$$

Bob can easily decrypt the ciphertext using (4).

$$m = C_1^{-k} \times C_2(mod\, n) \qquad (4)$$

This encryption scheme is homomorphic with respect to multiplication as if (x_1, y_1) and (x_2, y_2) are valid encryptions for messages m_1 and m_2, with the same key, then $(x_1 x_2, y_1 y_2)$ is a valid encryption of $m_1 m_2$. Hu *et al.* proposed a simple modification to make ElGamal additively homomorphic by placing the message m in the exponent [24]. So, if we encrypt two messages m_1 and m_2 using (3) but multiply h^l with g^m instead of m, the multiplication of the two ciphertexts results in a valid encryption of $g^{m_1+m_2}$. The problem here is that recovering the message involves solving a discrete logarithm problem (DLP) and this is precisely the problem whose difficulty ensures security. To solve this problem, they introduced a new scheme, called CRT-based ElGamal Scheme (CEG), which uses the Chinese Remainder Theorem (CRT) to replace one DLP in a large space by several similar problems in a more tractable search space. This allows for easily obtaining $m_1 + m_2$, while retaining the full security of the scheme, as shown later.

C. CRT-based ElGamal (CEG) Scheme

To illustrate how CEG works, let us reuse the previous example of *Alice* and *Bob*. In the first step, *Bob* also chooses a secret random exponent k and a generator g. He also chooses d_i for $i = 1, \ldots, t$ such that $gcd(d_i, d_j) = 1$ for $i \neq j$. So, his public key is $(g, h, (d_1, \ldots, d_t))$, where $h = g^k(mod\, n)$ and n is a large prime. For encryption, *Alice* sends the encryption of message m as a t-tuple of pairs (C_1, C_2) by using (5).

$$C_1 = g^{l_i}(mod\, n), C_2 = h^{l_i} \times g^{m_i}(mod\, n) \qquad (5)$$

where $m_i = m \ (mod \ d_i)$ and l_i is a generated random exponent for $i = 1, \ldots, t$. *Bob* can decrypt the ciphertext using (6) and (7).

$$m = CRT^{-1}[(log_g(C_{2_i} \times C_{1_i}^{-k}(mod\, n)), i = 1, \ldots, t)] \quad (6)$$

$$CRT^{-1}[C_i] = \sum_{i=1}^{t} C_i \frac{d}{d_i}(\frac{d}{d_i}^{-1} mod\, d_i)\, mod\, d \qquad (7)$$

Correctness and efficiency of the illustrated scheme is discussed in details in [24]. As a part of this work, we implement the CEG scheme in hardware and show its resource utilization and power consumption.

IV. HARDWARE TROJAN PROTECTION USING PH

Here, we introduce our suggested methods for defeating Hardware Trojan in third party IPs. First, we propose two schemes that support PH for the third party IP, which performs one type of operation (multiplication only or addition only). Then, we combine the two methods in a dual-circuit design that supports both multiplication and addition to satisfy applications that utilize the two operations.

A. Sufficient PH Support

Upon classifying the IPs based on processing type, one concludes that there is no need to afford the high cost of FH if the third party IP does only one type of operation. It is totally sufficient to have PH encryption/decryption before/after the suspected IP. In other words, if the suspected IP is used in an electronic voting system and only does addition operation to count votes on the server side [8], it is enough to support one of the additively homomorphic schemes mentioned before in Subsection III-A. For non computational suspected IPs, it is adequate to do simple obfuscation functions before the suspected IP and do the inverse of that function afterwords. Here, we discuss two partial homomorphic hardware implementations based on ElGamal encryption scheme described in Subsection III-B. The first implementation is the main ElGamal encryption/decryption scheme [23], which is a multiplicative homomorphic scheme. The second one is the CEG scheme [24], which is an additive homomorphic scheme.

1) Elgamal Scheme Implementation: Fig. 2 shows the block diagram for our implementation of ElGamal encryption/decryption scheme. The encryption module consists of two Montgomery modular multipliers, two Montgomery modular exponentiators, and a finite state machine (FSM) controller that is responsible for synchronizing other components' inputs and outputs to perform the encryption operations defined in (3). The decryption module consists of one Montgomery modular exponentiator, one modular divider, and a FSM controller that is also responsible for synchronizing other components' inputs and outputs to perform the decryption operations defined in (4). Both modules use a clock and reset signals as inputs. Reset and done signals are utilized to indicate the start and the end of module operations. The message m, ciphertexts $C_1 \ and \ C_2$, and the public key h are all k bits vectors, where k is a user-defined integer.

Montgomery multipliers were used in the design as the Montgomery's algorithm [25] is the most widely used algorithm for efficient modular multiplication. Other multiplication methods like the *multiply and reduce* and *double, add, and reduce* are computationally more complex [26]. The binary Montgomery multiplier employs only simple addition, subtraction, and shift operation to avoid trial division, which is a critical and time-consuming operation in conventional modular multiplication. In fact, this multiplier computes $Z = X \times Y \times R^{-1} mod\, M$ instead of $Z = X \times Y mod\, M$, where R is a chosen integer that should be a power of two and relatively prime to M. So, in this case, the operands need to
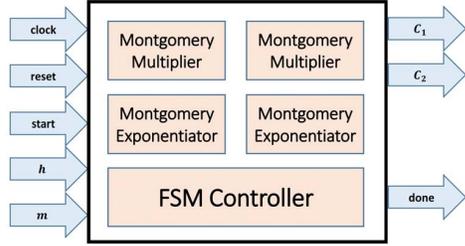
(a) Encryption.
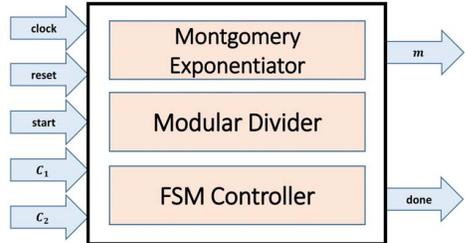
(b) Decryption.

Fig. 2: Block diagram for ElGamal encryption/decryption scheme.

(a) Encryption.

(b) Decryption.

Fig. 3: Block diagram for the CRT-based ElGamal (CEG) encryption/decryption scheme.

be converted into and out of Montgomery's domain each time this multiplier is used.

In general, modular exponentiation is usually accomplished by performing repeated modular multiplications. For our modular exponentiators, the LSB-first algorithm using Montgomery multiplication is used. This algorithm computes $Z = Y^X \bmod M$ in k executions of a loop that, in turn, includes at most two Montgomery multiplication operations, which are executed concurrently. That improves the performance of the module [26].

The decryption part of the scheme includes the usage of a modular divider module. We implemented the plus-minus algorithm as it gives the shortest computation time with a cost-effective area [27]. The key generation module consists mainly of a Montgomery exponentiation circuit and a true random number generator (TRNG) module, which is not in the scope of this paper. Finally, it is worth noting that the usage of only one multiplier and one exponentiator is enough to achieve the desired encryption results, but that results in a high critical path delay.

2) CEG Scheme Implementation: Fig. 3 shows the block diagram for our implementation of the CEG encryption/decryption scheme. This design is quietly different from ElGamal design discussed before as the encryption operations defined in (5) requires the usage of multiple Montgomery exponentiators. As the timing delay needed by one exponentiator is more than the delay of a single multiplier, the FSM controller is modified to utilize only one Montgomery multiplier. A modular reducer circuit is used to handle the operation of reducing m into several m_i based on the relation of $m_i = m(\bmod d_i)$ for $i = 1, \ldots, t$.

For the decryption module, it consists of one Montgomery
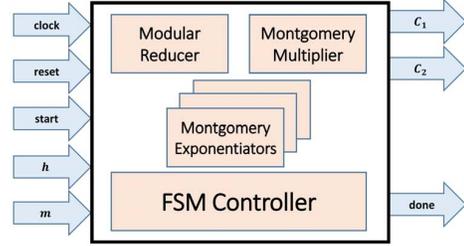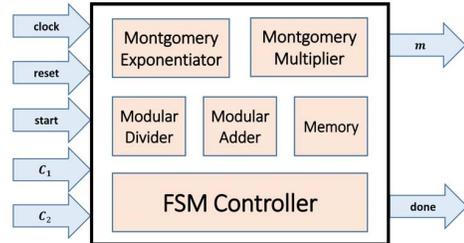
modular exponentiator, one Montgomery modular multiplier, one modular divider, one modular adder, FSM controller, and a single block of memory used to facilitate the implementation of the inverse CRT needed in (6) [24]. Input and output vectors are now $k \times t \; bits$ instead of $k \; bits$, where t is the number of ciphertext pairs.

B. Dual-Circuit Design

The main motivation for this design is that some third party IPs require the usage of more than one single type of operation. For instance, an IP may need to perform both addition and multiplication but not at the same time. One can imagine the functionality of that IP as a simple ALU that uses a selection line to switch its mode between two different operations. In this case, using one type of partial homomorphic schemes would not be sufficient. We have to implement two different schemes, such as implementing the two schemes described above, in order to prevent the attacker from revealing the ALU input and output data. We suggest a solution for this issue by combining the two previously proposed schemes, ElGamal and the CEG, in a single dual-circuit design. Thus, the proposed design supports both additive and multiplicative homomorphism.

Furthermore, we try to share resources as much as we can between the two schemes in order to have minimal design cost. For example, computing C_1 in (3) and (5) needs an exponentiation operation. The same situation occurs when computing C_2 as we need an exponentiation operation followed by a multiplication operation. The only difference is that the modified versions in (5) reuse their modules many times based on the value of t. Thus, we use the duality concept that enables us of sharing as much resources between the two circuits in

order to reduce the design area. As the CEG scheme uses the same basic blocks of ElGamal scheme with some additional blocks, we depend on the same architecture shown in Fig. 3 and add a *select* signal that chooses between the multiplicative homomorphic one and the additive homomorphic algorithms. The FSM controller is modified to be able of handling the two cases with the same building modules. The case is the same for the key generation and decryption modules.

By using this simple idea, we manage to decrease the area cost a lot and allow for the two homomorphic properties to be available on a single module. That completely solves the issue of the third party IP, which needs to perform both addition and multiplication operation. Moreover, another possible example for an application that needs the availability of both homomorphic operations is when we have two unique IPs in a design and the first IP performs addition while the second IP performs multiplication. Assuming that both IPs will not work on the same time, one can instantiate only one instance of our dual-circuit module and control its functionality to perform the needed operation of any of the two IPs, when needed, with the minimal cost in area and power consumption.

V. EXPERIMENTAL RESULTS

This section evaluates the performance of our proposed methods, described in Section IV, in terms of resource utilization, delay, and power consumption. The proposed methods are implemented on Xilinx Spartan-6 XC6SLX75 with FGG484 package and -2 speed grade. The area and performance results are obtained from the Xilinx ISE 14.6 tool after place and route analysis. The power is calculated using Xilinx Xpower Analyzer with 100 MHz clock.

A. PH Schemes Results

Table I shows the top-level module resource utilization of our two partial homomorphic encryption/decryption schemes, ElGamal and CEG, using vectors of size equals 8 bits.

Table II shows the maximum operating frequency of the two proposed partial homomorphic schemes along with the needed number of cycles to finish their work.

TABLE I: Resource utilization of ElGamal and CRT-based ElGamal (CEG) encryption/decryption schemes for k = 8 bits.

	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

TABLE II: Timing performance of ElGamal and CRT-based ElGamal (CEG) encryption/decryption schemes for k = 8 bits.

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

TABLE III: Power consumption (mW) of ElGamal and CEG encryption/decryption schemes for k = 8 bits.

	Encryption		Decryption	
	ElGamal	CEG	ElGamal	CEG
Clocks	5.65	7.87	4.21	5.87
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
Leakage	65.00	65.00	64.00	64.00
Total	93.82	92.02	78.15	80.27

From power prospective, Table III shows the power analysis for ElGamal encryption/decryption scheme and the CRT-based one. It was found that the dynamic power slightly decreased in case of encryption and increased in case of decryption due to the usage of the memory component and its logic controller in decryption. The leakage power remains constant in the both cases.

B. Dual-Circuit Design Results

Here, we compare the results of our proposed dual-circuit design to using regular two IPs, one for ElGamal and another for CEG design without any resource sharing between them. We want to address the effect of our resource sharing. In order to differentiate between the two designs, we call the first design, *Dual ElGamal*, while the second design is called *Regular ElGamal*.

Firstly, Table IV shows the area reduction that results from using our *Dual ElGamal* design over *Regular ElGamal* design. The area reduction column is calculated using (8). It is clear that the idea of dual-circuit design has greatly improved the usage of hardware resources.

$$Reduction(\%) = \frac{Regular\ area - Dual\ area}{Regular\ area} \times 100. \quad (8)$$

Table V gives the maximum operating frequency of our *Dual ElGamal* design and the *Regular ElGamal* design using vectors of size $k = 8\ bits$. The number of cycles here represents the clock cycles needed to perform one multiplicative homomorphic operation followed by one additive homomorphic operation. The needed number of cycles to get the final output is the same in both designs, except that the encryption part of our dual designs utilizes more clock cycles.

TABLE IV: Area reduction of our Dual ElGamal design over the Regular ElGamal design for k = 8 bits.

	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

TABLE V: Timing comparisons between our Dual ElGamal design and the Regular ElGamal design for k = 8 bits.

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

TABLE VI: Power consumption (mW) of our Dual ElGamal design and the Regular ElGamal design for k = 8 bits.

	Encryption		Decryption	
	Regular	Dual	Regular	Dual
Clocks	11.78	6.89	8.78	4.86
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
Leakage	65.00	65.00	65.00	64.00
Total	119.44	95.03	91.77	80.52

That is due to the usage of only one Montgomery multiplier instead of two, as illustrated in Section IV.

From power prospective, Table VI shows the power analysis for our *Dual ElGamal* design and the *Regular ElGamal* design. The usage of the duality idea results in an obvious improvement in total power consumption as it eliminates the power consumed by the duplicated modules. The savings in power consumption are 20.44% for encryption and 12.26% for decryption.

VI. CONCLUSION

In this work, we highlighted the importance of homomorphic encryption in defeating Hardware Trojans in third party IPs. As PH is sufficient enough with some third party IPs, we implemented two designs that supports PH (multiplicative only and additive only) based on ElGamal encryption/decryption scheme.

Furthermore, we integrated the two designs together and introduced a dual-circuit design that achieved a great improvement in area and power over a regular design that combines two IPs, one for ElGamal and another for CEG, without any resource sharing between them. Our architectures were implemented on a low-cost Xilinx Spartan-6 FPGA and area, delay, and power results were reported.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Design & Test of Computers*, 27(1):10–25, January 2010.

[2] Y. Gahi, M. Guennoun, Z. Guennoun, and K. El-Khatib. An Encrypted Trust-based Routing Protocol. In *IEEE Conference on Open Systems (ICOS), 2012*, October 2012.

[3] D. Hrestak and S. Picek. Homomorphic Encryption in the Cloud. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014*, pages 1400–1404, May 2014.

[4] H. Salmani, M. Tehranipoor, and J. Plusquellic. New design strategy for improving hardware Trojan detection and reducing Trojan activation time. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, HOST '09, pages 66–73, July 2009.

[5] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri. Design and analysis of ring oscillator based Design-for-Trust technique. In *VLSI Test Symposium (VTS), 2011 IEEE 29th*, pages 105–110, May 2011.

[6] A. Al-Anwar, Y. Alkabani, M.W. El-Kharashi, and H. Bedour. Defeating hardware spyware in third party IPs. In *Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, April 2013.

[7] A. Al-Anwar, Y. Alkabani, M.W. El-Kharashi, and H. Bedour. Hardware Trojan Protection for Third Party IPs on FPGA. In *16th EUROMICRO Conference on Digital System Design*, pages 662–665, Santander, Spain, September 2013.

[8] M. Tarek Ibn Ziad, A Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour. E-Voting Attacks and Countermeasures. In *28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)A-2014)*, pages 269–274, Victoria, BC, Canada, May 2014.

[9] A Al-Anwar, Y. Alkabani, M.W. El-Kharashi, and H. Bedour. Hardware Trojan detection methodology for FPGA. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 177–182, Victoria, BC, Canada, August 2013.

[10] R. Rad, J. Plusquellic, and M. Tehranipoor. A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions. *IEEE Trans. Very Large Scale Integr. Syst.*, 18:1735–1744, December 2010.

[11] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 51–57, June 2008.

[12] S. Wei and M. Potkonjak. Scalable Hardware Trojan Diagnosis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(6):1049–1057, June 2012.

[13] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran, and K. Rosenfeld. Trustworthy Hardware: Trojan Detection and Design-for-Trust Challenges. *Computer*, 44(7):66–74, July 2011.

[14] X. Zhang and M. Tehranipoor. Case study: Detecting Hardware Trojans in Third-party Digital IP Cores. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 67–70, San Diego, CA, USA, June 2011.

[15] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design & Test of Computers*, 27(1):66–75, January 2010.

[16] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009.

[17] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss. On the Design of Hardware Building Blocks for Modern Lattice-based Encryption Schemes. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'12, pages 512–529, Leuven, Belgium, 2012.

[18] T. Pöppelmann and T. Güneysu. Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware. In *Progress in Cryptology - LATINCRYPT 2012*, volume 7533 of *Lecture Notes in Computer Science*, pages 139–158. 2012.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[20] J. B. Clarkson. Dense Probabilistic Encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[21] P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, New York, USA, 1999.

[22] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[23] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. 1985.

[24] Y. Hu, W. J. Martin, and B. Sunar. Enhanced Flexibility for Homomorphic Encryption Schemes via CRT. In *International Conference on Applied Cryptography and Network Security, ACNS*, pages 93–110, Singapore, June 2012.

[25] P. L. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44(170):519–521, 1985.

[26] J. P. Deschamps. *Hardware Implementation of Finite-Field Arithmetic*. McGraw-Hill, Inc., New York, USA, 2009.

[27] J. P. Deschamps and G. Sutter. Hardware Implementation of Finite-Field Division. *Acta Applicandae Mathematica*, 93(1-3):119–147, 2006.