Hardware-Based Attacks to Compromise the Cryptographic Security of an Election System

Mohammad-Mahdi Bidmeshki, Gaurav Rajavendra Reddy, Liwei Zhou, Jeyavijayan Rajendran and Yiorgos Makris
Electrical Engineering Department, The University of Texas at Dallas
Richardson, TX 75080, USA
Email:{bidmeshki, gaurav.reddy, lxz100320, jv.ee, yiorgos.makris}@utdallas.edu

***Abstract*—We present our experiences in implementing hardware-based attacks to subvert the results of an election system. The election system was outlined by the Cyber Security Awareness Week (CSAW) Embedded Security Challenge (ESC) competition in 2015, held at the New York University (NYU). The system had multiple layers of security and primarily used homomorphic encryption. The competition presented a challenge to hack the election system such that a preferred candidate wins the election. We cryptanalyzed the given election system to evaluate the effectiveness of various theoretical and practical attacks, and used a custom designed embedded system to demonstrate our attacks. The embedded system was implemented on a Nexys 4 DDR Artix-7 FPGA board. Our work, which earned the first place in the competition, demonstrates that low-cost hardware-based attacks can indeed lead to catastrophic consequences.**

I. Introduction

The global distribution of the electronics supply chain provides ample scope for adversaries to insert hardware Trojans or backdoors into integrated circuits (ICs). Such malicious modifications can be made in the original design of intellectual property (IP) cores before they are distributed to customers and can be exploited later to break the security of a system.

To demonstrate the seriousness of the threat of hardware Trojans, NYU's ESC hosted a contest in November of 2015, wherein competitors played the role of attackers and were asked to overthrow a hypothetical election by altering the results in their own favor. The competitors would design and implement a hardware Trojan-infested terminal using an FPGA in order to launch a real attack on the cryptosystem. Correctness, performance and novelty were the grading criteria.

In this paper, we cryptanalyze the encryption scheme, explain the weak points identified in the system, and also describe the design and implementation details of our attacks. We exploited many of the weaknesses in the defined scheme, we developed effective Trojans with several optimization techniques to ensure efficiency, and we were the only successful team in breaking the election system and demonstrating the attacks in hardware [1].

The remainder of the paper is structured as follows: Section II briefly describes the baseline cryptosystem. Section III provides detailed analysis of the vulnerabilities in the system and the ways in which they are exploited. Section IV explains the implemented Trojan-infested embedded system with its optimizations. Section V describes the experimental setup and evaluates the performance of the hardware implementation in terms of power, area, and speed. Section VI briefly describes the theoretical optimization formulation to minimize the number of data frames needed for a successful attack, and Section VII concludes the paper.

II. Background

The provided election system in the competition comprises of several terminals for taking votes and a tally server, connected through Ethernet. It is assumed that messages are broadcasted in the network and that the rogue terminal can listen to all the data frames and can send malicious frames to the tally server to overturn the election result.

This system is secured by a cryptoprocessor that implements authenticated homomorphic encryption based on Paillier cryptosystem [2] and fast Message Authentication Code (MAC) over a sequence of concatenated ciphertexts, following the Encrypt-then-MAC paradigm. The cryptoprocessor is installed on the central tally server as well as on each election terminal, so that votes for each candidate are encrypted under a public key and paired with the candidate unique ID, before the unordered sequence of pairs is MAC-ed under a factory installed secret key. These pairs and the MAC are sent via broadcasted Ethernet frames to the tally server for homomorphic accumulation and verification of the MAC [3].

Paillier's cryptosystem is a homomorphic public key cryptosystem. Generating the public and private keys starts from choosing two large-enough prime numbers, namely p and q, with similar bit-length. Given p and q, a simple method of generating the public key used in this election system is to set $n = pq$ and $g = n + 1$. The private key is also generated from p and q, and requires multiplication, division and modular inverse operations [2], [4]–[7]. Breaking the cryptosystem and revealing the private key requires finding the prime factors of n, which is generally a very hard problem.

The fast MAC generation uses padded Merkle-Damgard chaining algorithm, where the hash function is as follows: $h(h', m') = (((((h' << 13) \mid\mid m') \ mod \ 524287) + (h' \times (h' >> 5)) \oplus (m' << 7)) \mod 65536$. In this function, h' is the chaining variable and m' is a 16-bit message block. MAC generation employs a secret key as the initial vector and the rogue terminal should be able to break the MAC in order to send authenticated data frames to the tally server.

978-1-5090-5142-7/16/$31.00

Finally, an integrity check is performed on the total number of votes. The tally server is aware of the total votes and does not approve results if the total votes go over this known value.

III. System Vulnerabilities

In this section, we briefly describe various vulnerabilities existing in the provided system and how we exploit them to stage our attacks.

A. Weak Key-Pairs in Paillier's Key Generation

The attack model outlined by the competition allows us to assume that a Hardware Trojan has been inserted by an adversary into the key generation core, which reduces the bit-length of the two prime numbers p and q from 512 bits to 16-28 bits [3], leading to the generation of weak key-pairs. Such hardware attacks which reduce randomness are not unrealistic, as demonstrated in [8]. We exploit this vulnerability by finding the prime factors of n using our rogue terminal and computing the values of p and q to generate the private key as explained in Section IV-A.

B. Homomorphic Addition of Votes

The Paillier cryptosystem is an additive homomorphic cryptosystem in which, given the public-key and the ciphertext of messages m_1 and m_2, one can compute the ciphertext of $m_1 + m_2$ without decrypting the messages. The tally server uses this property and therefore, it does not check the integrity of individual counts it receives for each candidate. Since there is a final modulo n operation in Paillier's decryption, the rogue terminal can send encrypted values of $n-k$ to reduce the tally for a candidate by k. We exploit this vulnerability to modify vote counts of candidates without affecting the total number of votes.

C. Merkle-Damgard Construction of MAC

The padded Merkle-Damgard (MD) construction used in MAC generation splits input message into blocks and then processes them iteratively, as shown in Fig. 1. We implemented two types of attacks to generate valid MAC for a frame without knowing the secret initial vector.

Length extension attack: It works by appending additional data to the end of a frame and generating a legitimate MAC for this new frame, considering the MAC of the original frame as the initial vector. This is possible since the defined data frame structure allows having dummy bytes which are discarded in data frame interpretation. Although length extension attack is efficient in generating a legitimate MAC, care must be taken on padding and frame length in MAC generation. Also, the vote count in the original frame is sent again, which must be considered in tracking the current vote count on the tally server. Additionally, the tally server only reads the first vote count for a candidate in a frame and therefore, selecting a received frame to append additional information requires special attention. We formulated the problem of selecting frames and determining the number of votes needed to alter the results, as an integer linear programming (ILP) problem, described briefly in Section VI.

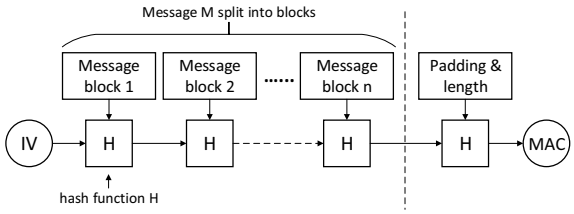

Fig. 1: The Merkle-Damgard construction

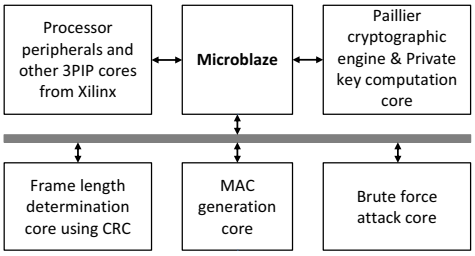

Fig. 2: Block diagram of the system primarily indicating the custom cores implemented in hardware

Brute-force attack: Due to rather small bit-length of blocks and the initial vector (16 bits), a brute-force attack is also possible, where we try to find the initial vector by examining all possible values and checking the final MAC on a data frame. Compared to the length extension attack, a brute-force attack is less efficient but provides free manipulation of the frame that an attacker can create and send. In our rogue terminal, we implemented both attacks and employed them where appropriate, as described in Sections IV-C and IV-D.

IV. Hardware Implementation

Fig. 2 shows the block diagram of the embedded system of our rogue terminal. The custom cores implemented in the hardware are: a) Private key computation core, b) Paillier cryptographic engine, c) MAC brute-force attack core, d) MAC generation core with length extension attack, and e) Cyclic Redundancy Check (CRC) core. The Ethernet controller, memory interface and other processor peripherals are implemented using IP cores publicly available from Xilinx.

A. Private Key Computation Core

To find the private key of the tally server, first we need to find the prime factors of the public key, which is available in every transmitted frame. By extracting the public key from the first frame received, we use the following procedure to find the prime factors (p and q) of the server's public key, namely n. Due to the limited bit-length of p and q (16 to 28 bits), we use exhaustive search for this purpose. Since $n = pq$, p and q have the same bit-length, and we have the bit-length of n, we can get the approximate bit-length of p and q by dividing the bit-length of n by 2, which in turn reduces the search space.

In the hardware implementation, we employ a pipelined divider which computes a division every two clock cycles. Upon receiving n (the public key), p, and q are initialized with the first/last potential prime numbers with our approximated bit-length. On every cycle, p and q are increased/decreased by

2. If their current values are not a multiple of 3, 5, 7, 11, and 13, we send them to the divider to compute the result of n/p (or n/q). When we get a remainder of zero, the private key is found. To increase the performance, we employed 12 such cores which start from different potential values for p and q, reducing the average time required to find the private key by a factor of 12, as compared to a single core.

B. Paillier's Cryptographic Engine

Paillier uses several modulo multiplication and modulo exponentiation operations in encryption and decryption. We leveraged single cycle multipliers provided by the hardware platform and implemented these modulo operations based on Montgomery multiplication and exponentiation algorithms [9]. Montgomery only requires multiplication and shift operations and removes the need for high cost dividers, which leads to higher performance of our crypto core.

For computing multiplication and exponentiation modulo n or n^2 used in Paillier cryptosystem where n is the public key, Montgomery algorithm requires a few precomputed parameters based on n. We start computing all these parameters once we receive the first frame carrying the public key. When the prime factors become available, the private key is computed and the crypto core becomes ready to encrypt/decrypt messages. The inverse modulo used in the cryptosystem is implemented in hardware using a binary GCD algorithm [9].

C. MAC Brute-force Attack Core

This component implements the capability to launch a brute-force attack to find the IV header and the factory key used for MAC generation. A complete brute-force attack computes the MAC on one frame with all possible *headers* exhaustively and finds whether there is a match in the MAC value. A subset of possible headers will be generated accordingly. The subset can then be applied on the next frame with the same computation process to generate a new subset. This operation can be executed repeatedly until a single header value is found.

We performed several optimizations in hardware implementation to improve its performance. In the hash function used in the MAC generation, the modulo operation of a large prime number, $2^{19}-1$, is included, increasing the complexity of the hardware design. Thanks to the work in [10], a modulo 2^n-1 operation can be reduced as

$$A \; mod \; (2^n-1) = (A \; mod \; 2^n + A \; div \; 2^n) \; mod \; (2^n-1) \quad (1)$$

and therefore can be linearized. Moreover, 32 parallel computation units are allocated in this core in order to enhance the computation speed. Eventually, the actual running time is bounded by $O(2^{11} \times number \; of \; blocks \; in \; one \; frame)$.

D. MAC Generation Core

This component implements the MAC generation algorithm using MD construction and also enables the length extension attack. Two operation modes, i.e. normal mode and attack mode, are introduced in this component. The normal mode computes the MAC for a new frame in the brute-force attack.

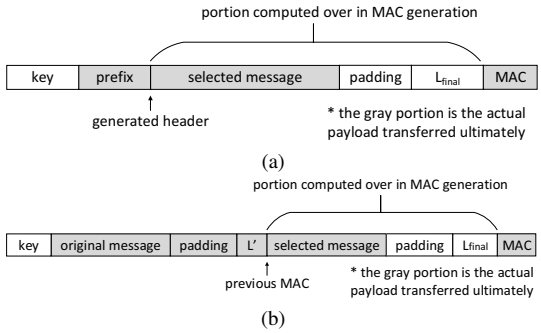

Fig. 3: Frame format in normal mode with normal functionality vs. attack mode under length extension attack

The attack mode, on the other hand, generates the MAC for frames modified by the length extension attack. The format of the generated frame in both modes is shown in Fig. 3.

Similar to the brute-force attack core, hash function computation is optimized. This attack is expected not to add any runtime overhead since the MAC generation procedure must be called before sending any frame even in a legitimate terminal.

E. CRC Core

The Ethernet core does not provide any information about the length of the received frame and this makes handling and interpretation of received raw Ethernet frames difficult. To resolve this issue, we find the length of the frame by computing its CRC, including received CRC bytes. Once CRC matches a specific *magic residue*, we know the end of the frame is reached [11]. To improve performance, we implemented the CRC computation in hardware.

F. System Work Flow

After initializing the hardware cores, the system waits to receive data frames from Ethernet. Upon receipt of the first frame, the public key is recovered and the key-finding core starts to find its prime factors. Also, the MAC brute-force core starts to recover the initial vector for MAC generation. In the mean time, received data frames are queued for processing. Once the private key is found, vote counts of candidates are decrypted and tallied. Once the contents of all received packets are processed, depending on whether the initial vector for MAC generation is found or not, we send appropriate extended data frames or new frames to the tally server to ensure the favorite candidate wins the election. For length extension, we use heuristic algorithms to select best data frames to append new data to, or keep the best set of original frames for future use. Once the initial vector is found by the MAC brute-force core, the length extension attack is no longer needed. The system continuously monitors the incoming data frames and sends new frames to keep the preferred candidate in the lead.

V. Results & performance evaluation

A. Experimental setup

The rogue terminal was implemented using HDL on a Nexys 4 DDR Artix-7 FPGA board, where the software program was executed on a Microblaze processor and coordinated with hardware components to launch attacks [12].

TABLE I: Post-implementation area, power, and performance summary

Module	Area[a]		On-chip power in (mW)	Speed in (Mhz)
	Slice LUTs	Slice registers		
Paillier crypto engine and private key computation core	31851 (60.4%)	43036 (69.4%)	369 (25.7%)	25 (Crypto), 75 (P. key)
Brute-force attack core	6844 (10.7%)	5140 (8.3%)	91 (6.3%)	75
Microblaze	1918 (3.6%)	1759 (2.8%)	37 (2.6%)	75
Frame length determination core using CRC	205 (0.003%)	126 (0.002%)	1 (<0.07%)	75
MAC generation core	193 (0.003%)	181 (0.003%)	2 (<0.14%)	75
Processor peripherals and 3PIP cores from Xilinx	22389 (22.10%)	11735 (18.9%)	934 (65.19%)	50 - 200
Complete Design	(52649 of 63400) (83%)[b]	(61977 of 126800) (48.9%)[b]	(1434 of 2178) (66%)[c]	25-200 (Internal clocks)

(a) % of the resources used by the complete design. (b) % of the total FPGA resources used. (c) % of the total power dissipated by the board.

Xilinx Vivado design suite and Xilinx Software Development Kit (SDK) were used for design and implementation. Custom scripts were used to generate raw Ethernet frames and Wireshark [13] was used to monitor the frame transmission. After the bitstream was generated and programmed into the FPGA board, the operation of the rogue terminal was tested in a simulation environment.

B. Performance Evaluation

The design characteristics are evaluated in terms of power, area and clock frequency. A post-implementation performance summary is shown in Table I.

VI. Optimizing Length Extension Attack Using ILP

The MAC algorithm used in this voting system is vulnerable to length extension attacks. This enables us to create verifiable MAC for those messages which are collected from a legitimate source and modified by adding additional data to the end of the message. Therefore, to perform length extension attack on the system, we need to store the original messages coming from legitimate terminals in the network, decrypt their contents to know how they affect the tally, add new data to manipulate the vote count and send the extended message to the server.

Despite this attack possibility, there are some restrictions. For instance, the tally server does not accept a vote count appearing twice for the same candidate within the same data frame. Furthermore, as described in Section III-C, sending an extended frame to the tally server causes the vote count in the original frame to be counted twice, and those votes should also be compensated for by either altering the votes of other candidates within this frame or by sending additional frames after the current frame. Therefore, in order to manipulate the votes of a specific candidate more easily, we need to find the most appropriate frame which can best serve our purpose.

Although we implemented the length extension attack in our rogue terminal based on heuristic algorithms, this implementation is not necessarily optimal. In our system, this is not a major problem because the moment we find the initial vector for MAC generation, we no longer need the length extension attack. However, to provide more insight, we formulated this problem as an ILP optimization. This formulation takes into account two cases of overturning the election using only the original messages with and without extended content, and considers all the constraints of the problem such as limited length of data frames, pre-existing votes in the frame, etc. Due to space limitations, we omitted the details of this formulation. The complete formulation can be found in the detailed report [14].

VII. Conclusion

We evaluated the security of an election system outlined by the NYU CSAW ESC-2015 competition. We cryptanalyzed the system, exploited its weak points and launched variety of hardware-based attacks to subvert the results of the election. All our attacks were launched using a rogue terminal built on the Nexys 4 DDR Artix-7 FPGA board. In this paper, we analyzed the vulnerabilities within various security layers of the homomorphic encryption based voting system and we demonstrated the catastrophic consequences of the presence of hardware Trojans/design-errors in cryptographic ICs.

References

[1] CSAW 2015 Finals. [Online]. Available: http://engineering.nyu.edu/press-releases/2015/11/16

[2] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptology*. Springer, 1999, pp. 223–238.

[3] [Online]. Available: https://github.com/nekt/csaw_esc_2015

[4] S. Choinyambuu. Homomorphic tallying with paillier cryptosystem. [Online]. Available: http://security.hsr.ch/msevote/seminar-papers/HS09_Homomorphic_Tallying_with_Paillier.pdf

[5] J. S. P.-A. Fouque, G. Poupard, "Sharing decryption in the context of voting or lotteries." Springer-Verlag, 2000, pp. 90–104.

[6] N. G. Tsoutsos and M. Maniatakos, "Heroic: homomorphically encrypted one instruction computer," in *Proceedings of the conference on Design, Automation & Test in Europe*, 2014, p. 246.

[7] O. Mazonka, N. Tsoutsos, and M. Maniatakos, "Cryptoleq: A heterogeneous abstract machine for encrypted and unencrypted computation," in *IEEE Transactions on Information Forensics and Security*, vol. 11, 2016, pp. 2123–2138.

[8] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *Proc. of Int. Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 197–214.

[9] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[10] R. Zimmermann, "Efficient vlsi implementation of modulo $2^n \pm 1$ addition and multiplication," in *Proc. of 14th IEEE Symp. on Computer Arithmetic*, 1999, pp. 158 – 167.

[11] Configurable locallink CRC reference design. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp562.pdf

[12] Microblaze processor reference guide. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug984-vivado-microblaze-ref.pdf

[13] Wireshark. [Online]. Available: https://www.wireshark.org/

[14] CSAW 2015 Report. [Online]. Available: http://utdallas.edu/%7Ebidmeshki/csaw15/trelareport.pdf