2014 28th International Conference on Advanced Information Networking and Applications Workshops

E-voting Attacks and Countermeasures

M. Tarek Ibn Ziad, Amr Al-Anwar, Yousra Alkabani, M. Watheq El-Kharashi, and Hassan Bedour
Department of Computer and Systems Engineering
Ain Shams University
Cairo 11517, Egypt
{mohamed.tarek, amr.alanwar, yousra.alkabani, watheq.elkharashi, hassan.bedour}@eng.asu.edu.eg

***Abstract*—Electronic voting (e-voting) systems have been in use since the 1960s. E-voting offers many advantages compared to other voting techniques. However, it also introduces many security challenges. As it may contain malicious back-doors that can affect system dependability. In this work, we present one of e-voting challenges where the hardware Trojan tampers results totally.**

We implement an e-voting machine as a case study on Xilinx FPGA board. Then, we inject a hardware Trojan to tamper voting results. The attack depends mainly on the unused bits. We provide a protection technique and show its overhead. Furthermore, we introduce other attacks and protection scenarios. We compare between our selected protection techniques and others techniques. Finally, we illustrate that our chosen protection technique incurs negligible power overhead, whereas the average area and delay overheads are 4% and 10%, respectively.

***Index Terms*—E-voting, hardware spyware, hardware Trojan, security.**

I. Introduction

Democracy depends on the election process to satisfy population needs. Elections give the advantages for the populace to choose their representatives and express their preferences. No one can deny that the integrity of the election process is very important to the integrity of democracy itself. Also, the election system must be sufficiently robust, transparent, and comprehensible that voters and candidates must accept the results of an election. On the other hand, a lot of election examples had been manipulated in order to influence their outcomes. The design of a good voting system must satisfy a number of some competing criteria.

E-voting is an election methodology in which the votes are checked or collected electronically. A computer system whose main element is a software component that maps the voting procedure electronically is called an e-voting system. E-voting systems offer many advantages for both the voters and election administrators as described in [1]. Moreover, voters seem to prefer electronic voting systems due to their privacy and accessibility.

E-voting security is one of the most important topics nowadays. The Caltech MIT Voting Technology Project [2] in their report state: "security is as important as reliability in guaranteeing the integrity of the voting process and public confidence in the system. People do not use things in which they have no confidence." In [3], Yumeng et al. reviewed the researches on the electronic voting schemes toward trustworthy voting system and discussed the issues about several important properties and challenges. Kumar and Begum discussed an introduction about Electronic Voting Machine (EVM) and its variation [4].

In [5], the authors presented a minimal implementation of a cryptographically secure electronic voting system, built with a low-cost Xilinx FPGA board. This system, called VoteBox Nano, follows the same basic design principles as VoteBox while restricting some network features so as to fit on a cheap FPGA. It is a very simple design running without any operating system. It only consists of an FPGA connected to an interface screen (VGA) and a keypad to allow the voter to choose his/her desired candidate and confirm his choice.

In this paper, we introduce a full e-voting system implementation. Then, we show a simple scenario for an untrusted machine and how it would be used to affect the election results. Furthermore, we introduce protection against the proposed Trojan. The rest of the paper is organized as follows. Section II presents the related work. Section III shows the original design specifications. In Section IV, a scenario for a possible attack is introduced. Methods of protection are presented in Section V. Other attacks and countermeasures are introduced in Section VI. Section VII discusses the experimental environment details. Section VIII contains the final results. Section IX concludes our work.

II. Related Work

There is no doubt that designing an election system needs special care. An electronic election should be more secure, transparent and trustworthy, as common people have less faith in computers due to hacking threats and system crashes. Kohno el al. discussed some of e-voting system problems such as incorrect use of cryptography, unauthorized privilege escalation, vulnerabilities to network threats, and poor software development processes [6].

In [7], Fauzia et al. describe an implementation of an efficient and secured electronic voting system based on the Fujioka- Okamoto-Ohta protocol. Their implementation contains the automation of an online voting system providing some features, which were not available in the previous implementations. Another design of an electronic voting machine was introduced by Alam et al. in [8]. Their main aim of the project was not to design power-efficient perfect device, but was to design a mother device that can be adopted to any recent technologies. Their machine also used the voter's ID to identify a valid voter and restrict multiple vote casting.

978-1-4799-2652-7/14 $31.00 © 2014 IEEE
DOI 10.1109/WAINA.2014.53

Oksuzoglu et al. in [5] built a simplified VoteBox-like system, which they call VoteBox Nano, using a Xilinx Spartan-3E 500 Starter Kit. Their implementation combined modules, such as Xilinx's MicroBlaze soft-CPU core, with custom logic for fast cryptography and for generating truly random numbers. The application of VoteBox Nano, itself, is written in C and runs on the MicroBlaze processor.

Wollinger et al. [9] provide a summary of security issues while doing cryptography on an FPGA, with a focus on how to maintain cryptographic secrets within the FPGA in the face of attacks like attempting to read out the FPGA's bitstream "readback attack", its internal SRAM, and so on. If the bitstream of the FPGA, itself, is a trade secret, then the ability to read it out could be sufficient to reverse-engineer the logic within it.

Xilinx and other FPGA manufacturers offer features aimed at preventing these reverse-engineering attacks [10]. For preventing "IP core theft," FPGA chips allow the bitstreams that define the FPGA configuration to be encrypted. When the FPGA boots, it can access an internal key store and use this to decrypt the bitstream. An invader reading the ciphertext would learn nothing and no queries are available to allow an invader to read the decryption key from the FPGA. Alternatively, Alkabani and Koushanfar [11] showed how to leverage chip-to-chip variations in their behavior to achieve *active hardware metering*. The FPGA configuration is made unique to a given chip; moving it to another chip would not yield a functioning implementation.

These techniques are aimed at protecting the secrecy of the FPGAs bitstream. For the VoteBox Nano, secrecy is not the issue. They need to detect tampering, which is a different problem. Drimer and Kuhn [12] described a protocol to enable an FPGA to reject configuration updates that are undesirable. Dutt and Li [13] proposed adding "parity groups" to the logic blocks within the FPGAs, so changes to any one logic block will cause parity failures without corresponding changes elsewhere, which the randomization makes difficult to defeat.

In case of obtaining machine components from third parties, design is exposed to further challenges that need to be faced. That include hardware spywares and hidden back-doors. Regular testing methods are not suitable to detect the faults caused by hardware Trojans because they are not expected to be activated during testing. This is because Trojan circuits are usually designed to be activated using a rare trigger. Recently, different methods have been specifically designed with Trojan detection in mind. We can classify these hardware Trojan detection methodologies into: (1) side-channel dependent methodologies and (2) architectural methodologies [14].

Side channel dependent methodologies try to localize the impact of the Trojan on the circuit without activating it. Jin and Makris use path delay analysis to detect Trojans [15]. Wang et. al use localized current analysis for Trojan detection. They measure power from multiple ports to detect the impact of the Trojan on power [16]. The impact of a Trojan on the power supply transient current of an IC is studied using statistical methods by Rad et al. [17]. Banga et al. introduced a test vector generation method that can be used to differentiate between the side-channel waveforms of a Trojan infected and a non-infected circuit [18], [19].

Architectural methodologies try to increase the chances of the activation of a hardware Trojan during testing. Banga and Hsiao use voltage inversion at alternating levels of the circuit to increase the power consumption of an infected circuit [20]. Salmani et. al increase the Trojan activity by inserting dummy flip flops in the design [21]. They choose the locations of the inserted flip flops based on a transition probability threshold.

The main problem of all these methods is that they require the presence of a non-infected (golden) chip. Thus, they are practical if the design does not contain third party IPs [22]. However, if the system designer inserts third party IPs in his design, these methods become less practical. Zhang and Tehranipoor try to provide an alternative to using a golden design by using formal verification, code coverage analysis, and ATPG methods to achieve high confidence in whether the circuit is Trojan-inserted or Trojan-free [23]. Baumgarten et al. introduced using reconfigurable logic barriers within a design to prevent the activation and operation of Hardware Trojans added during the manufacturing stage of an IC [24]. Waksman and Sethumadhavan introduced a method that attempts to prevent Trojan triggering [25]. Beaumont et al. ran replica of a program on multiple processing elements to achieve protection from hardware Trojans [26].

Al-Anwar et al. in [27] 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 use cyclic redundancy check (CRC) to detect the infected IP.

III. E-voting System Overview

The implemented e-voting system is similar to VoteBox Nano design [5]. Fig. 1 shows an overview of the full e-voting system. Voter logins to any of the e-voting boxes which are distributed over the country. Then, e-voting box encrypts the vote using El-Gamal encryption algorithm. Encrypted votes are sent to the main secured server via a network connecting the whole country holding the elections. The main server will receive the encrypted votes, decrypt them and collect the results together. The decryption methodology follows the same concept of El-Gamal Algorithm used in encrypting votes. Finally, voting results can be ready on even the same day without human interference.

Fig. 2 shows an abstraction of the implemented e-voting box. The true random number generator (TRNG) core block is responsible for generating keys, which are required for votes encrypting. TRNG depends mainly on post-processing of digitized noise. Every encrypted value in the system requires a distinct random number. We should highlight that choosing a TRNG algorithm is critical as numbers prediction may allow the attacker to decrypt the ciphertexts. Clearly, a voter's privacy relies on the unpredictability of the random numbers. We should also note that El-Gamal algorithm is an asymmetric

key encryption algorithm for public-key cryptography, which is based on the Diffie-Hellman key exchange [28]. Input keypad and VGA screen cores represent the input and output modules, respectively. The machine screen will display the names of the candidates with their numbers arranged from 1 to n. The voter will use the keypad buttons to select his candidate and confirm his choice. He/She can also control some other features, such as determining the screen brightness.

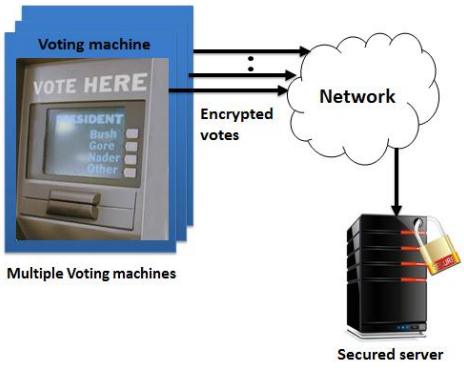

Fig. 1. The e-voting process.

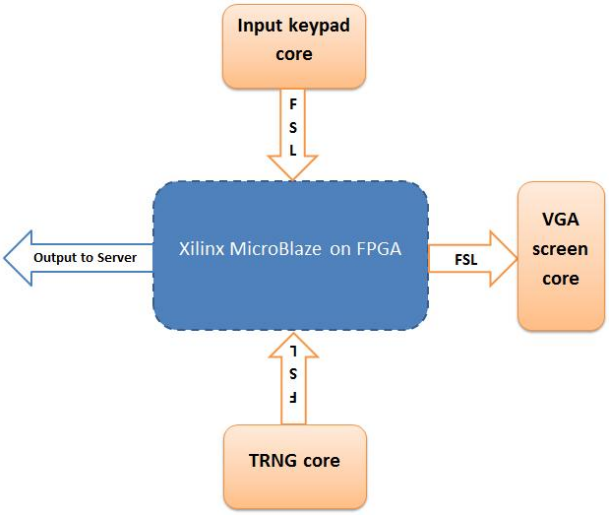

Fig. 2. Abstract view of the implemented e-voting machine.

Xilinx MicroBlaze is the main block in the design. It is a virtual microprocessor that is built by combining blocks of code called cores inside a Xilinx FPGA. The MicroBlaze processor is a 32-bit Harvard RISC architecture optimized for implementation on Xilinx FPGAs with separate 32 bit instruction and data buses running at full speed. That supports executing programs and access data from both on-chip and external memory at the same time. MicroBlaze is connected to other cores using the Fast Simplex Link (FSL). FSL is a uni-directional point-to-point communication channel bus that perform fast communication between any two design elements on the FPGA when implementing an interface to the FSL bus.

IV. Scenario for a possible attack

An untrusted FPGA-based voting machine may be used to tamper the legal votes of users. Attacking vendor may inject cores connected to the MicroBlaze via fast simplex link (FSL). These cores are responsible for dealing with inputs from the keypad and interfacing with the output screen. He may add a hidden core that replaces the user's vote with another one, if it receives a special external trigger. In our case study, we assume that the voting system contains a secret core connected to the MicroBlaze core and takes its input from the FSL coming from the input keypad core, as shown in Fig. 3.

In our experimental attack scenario, the thief could use the input keypad as follows. He/She will press the push button that control screen brightness with a secret sequence depending on the position of the wanted candidate. That secret sequence would be translated into data sent to the MicroBlaze via the FSL in the unused bits beside the regular data. As a result of triggering the MicroBlaze back-door, all the coming sent votes will support the wanted candidate whatever the voter chooses. Repeating that several times on several machines will affect the whole election results, significantly.

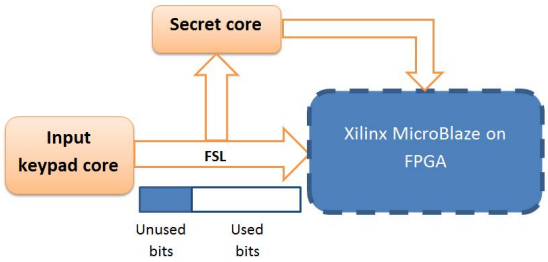

Fig. 3. Block diagram of a secret core for an untrusted e-voting machine.

V. Protection against proposed attack

There are several ways to protect our system from untrusted third party IPs. We suggest to use the *Simple Blockage (SB)* method introduced in [29] with a simple improvement. The authors suggest obfuscating the output of the suspected IP before sending out data, then undo 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 simple obfuscating function. We choose to protect using the simple function which is mainly based on data Xoring. Obfuscation will take place between keypad and MicroBlaze. Processing will be done, as shown in Fig. 5 and Fig. 6.

In our case, the data transmitted via the FSL is 32-bit. We enhance the *SB* method by resetting any unused bits to zero before receiving them at MicroBlaze. We will allow the trusted-known used bits only to go. Obfuscating the unused bits cost is wasted and will be omitted. Furthermore, an attacker may depend on the unused bits to discover our simple obfuscating function. We did not use the partial reconfiguration feature to change obfuscating function periodically as proposed in [29], as the partial reconfiguration feature doubles FPGA and e-voting box cost. Section VIII shows function overheads in

details. Our technique would prevent the trigger that would turn the secret core on.

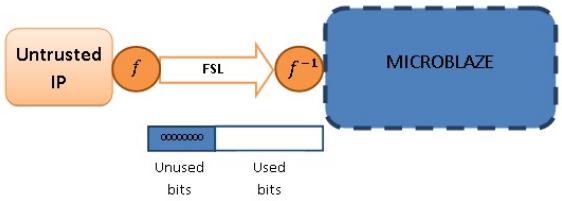

Fig. 4. E-voting protection against proposed attack.

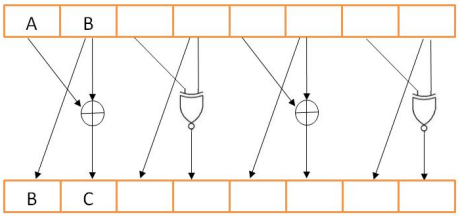

Fig. 5. Simple confusion function depending on data xoring.

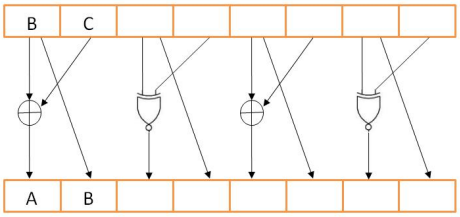

Fig. 6. Inverse of confusion function to retrieve data.

VI. Other Attacks and Countermeasures

In this section, we discuss some other possible attacks and propose countermeasures.

• **Sequence code attack.** In Section IV, we assumed that the secret core trigger will be sent to the MicroBlaze core in the unused bits in one data packet. That is called *single-shot cheat code* as described in [25].

Another possible attack may be based on sending a large cheat code in multiple pieces. That is called *sequence cheat code*. This type of cheat code arrives in small pieces over multiple cycles or multiple inputs. Just like the single-shot codes, these cheat codes can be supplied through the FSL data bus connecting the input keypad core and the MicroBlaze. In [25], Waksman and Sethumadhavan proposed two different ways to solve cheat code issues. One for the single-shot cheat code using data obfuscation (encrypt input values to untrusted units to prevent receiving special codes, thus preventing recognizing database triggers). Another way is to use sequence-breaking against sequence cheat code. The sequence breaking method suggests pseudo-randomly scramble the order of events entering untrusted units to prevent them from recognizing sequences of events that can serve as data-based triggers.

However, our proposed method in Section V is capable of protecting the design from that external trigger either if it is single-shot cheat code or even sequence cheat code. So, we do not have a need for extra hardware for sequence cheat code, and that is an advantage over Waksman methodology.

• **Used bits attack.** Now, let's introduce another attack where we suggest that keypad core is infected and no extra hidden cores. So, the trigger will be sent in the used bits. In all previous attacks, we assumed that the special trigger is sent in the unused bits of the data packet. The main risk is that the MicroBlaze sees that confidential data (user's vote) in unencrypted form and thus can manipulate it. Additionally, this core-and many other on-chip functional cores-are often procured as third party IP. In this case, using our technique to secure data transfer would not prevent triggering the hidden back-door because the trigger will be obfuscated at the output of the untrusted IP (input keypad core), transfer via the FSL and then return back to its original form at the input of the MicroBlaze core.

We should mention that Waksman [25] presented a solution for this attack by using data obfuscation for computational units. They suggested using the third party computational IP without giving it the advantage of recognizing the data. They depend mainly on homomorphic encryption schemes, as shown in (1). But, the main problem of this solution will be the overhead cost of all e-voting boxes.

$$Gamal(xy) = Gamal(x)Gamal(y) \tag{1}$$

If we want to encrypt a data value x, where x represents the vote in our case, using El-Gamal Algorithm on a special purpose encryption core, we can perform the following steps.

1) Use hardware to generate a random value y and calculate its encryption result $Gamal(y)$
2) Compute the product $z = xy$ using a regular, trusted ALU, where x is the value to be encrypted (user's vote).
3) Ship z off to the encryption core. That core returns $Gamal(z) = Gamal(xy) = Gamal(x)Gamal(y)$, which will be sent to the main server.
4) At the server side, add the received encrypted votes $Gamal(z)$.
5) Decrypt the summation result and use a trusted ALU on the server side to divide the result by y.

We have used the untrusted cryptographic unit in the e-voting box to encrypt the vote x without allowing this untrusted unit to know the actual value of x. This will protect from the triggering code injected in the real data.

VII. Experimental Setup

During this case study, the total experiment was done by using **Spartan 3e starter kit (XC3S500E-Device family, FG 320 package and -4 speed grade)**. The interface between cores is done by using Xilinx Platform Studio. The total application is developed by designing required logic in Verilog.

Fig. 7 illustrates the experiment setup. Our FPGA board is connected to a VGA screen via a VGA cable. The input keypad is connected to the FPGA using a PS2 interface.

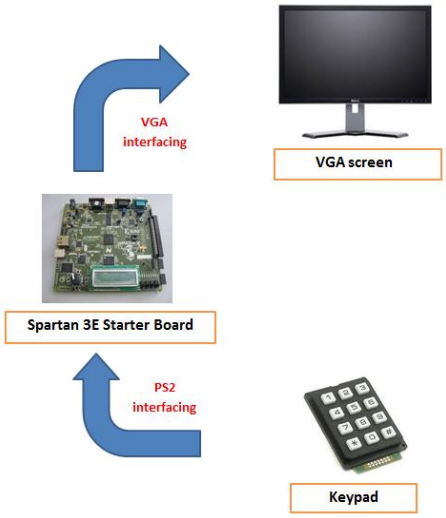

Fig. 7. Experimental setup to protect e-voting.

VIII. Experimental Results

In this section, we study the effect of using the protection technique described in Section V on device resources, power and time delays. Each time we make a comparison between results coming from the protected design and the original untrusted design. We use Xilinx Spartan 3e starter kit(XC3S500E-Device family, FG 320 package and -4 speed grade) to implement our designs. Furthermore, we analyze the overhead of resetting only the used bits in the first experiment. Then, we examine using the simple obfuscation function.

a) In case of resetting unused bits

Table I presents device utilization summary for the untrusted system and the protected one. The first column represents logic utilization. The second and third columns represent the original system data and the protected one, respectively. The last column shows the protection overhead.

TABLE I
Device utilization overhead with and without resetting unused bits.

	Without resetting unused bits	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

Also, power is calculated using Xilinx Xpower Analyzer with 50 MHz clock. Table II shows the power analysis for the untrusted system and the protected one. It was found that dynamic power decreased after inserting the protection method while the leakage power remains constant.

TABLE II
Power consumption (W) with and without resetting unused bits.

	Without resetting unused bits	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

From the timing prospective, we use the Post-PAR Static timing report generated from Xilinx Platform Studio v14.6 to get the design statistics. For the untrusted design, the minimum period is **16.757 ns** (max frequency: **59.677 MHz**). Moreover, the maximum net delay is **2.059 ns**. However, the minimum period is **15.846 ns** (max frequency: **63.107 MHz**) for the protected design. The maximum net delay is **2.265 ns**. So, delay overhead is **0.206 ns**.

b) In case of using enhanced SB method

We are using the enhanced simple obfuscation method which is described in Section V. Table III presents device utilization summary for the untrusted system and the protected one. The first column represents logic utilization. The second and third columns represent the original system data and the protected one, respectively. The last column shows the protection overhead.

TABLE III
Device utilization overhead with and without enhanced SB.

	Without enhanced SB	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

The power is calculated using Xilinx Xpower Analyzer with 50 MHz clock. Table IV shows the power analysis for the untrusted system and the protected one. It was found that total power decreased after inserting the protection method while the leakage power remains constant.

From the timing prospective, the untrusted design has a minimum period **16.757 ns** (max frequency: **59.677 MHz**), and a maximum net delay **2.059 ns**. However, For the protected

TABLE IV
POWER CONSUMPTION (W) WITH AND WITHOUT ENHANCED SB.

	Without enhanced SB	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

design, the minimum period is **19.737 ns** (max frequency: **50.666 MHz**). And, The maximum net delay is **2.264 ns**. So, the delay overhead is **0.205 ns**.

IX. CONCLUSION

In this work, we highlighted e-voting challenges. We implemented an e-voting machine as a case study on Xilinx FPGA board. Then, we injected a hardware Trojan in our e-voting machine to tamper voting results. The attack depends mainly on using the unused bits of the data sent from the input keypad core and the MicroBlaze main core. Moreover, we provided a protection technique to solve this issue and showed that it adds low delay and power overheads. The power overhead was negligible while the delay overhead did not exceed 10%. Device resources overheads did not exceed 4%.

REFERENCES

[1] S. Everett, K. Greene, M. Byrne, D. Wallach, K. Derr, D. Sandler, and T. Torous. Is newer always better? the usability of electronic voting machines versus traditional methods. In *CHI 2008*, Florence, Italy, 5-10 April 2008.

[2] V. T. Project. Voting: What is, what could be, report of the caltech MIT voting technology project. 2001.

[3] F. Yumeng, T. Liye, L. Fanbao, and G. Chong. Electronic voting: A review and taxonomy. In *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pages 912–917, Xi'an, China, 23-25 Aug. 2012.

[4] D. Kumar and T. Begum. Electronic voting machine–A review. In *Pattern Recognition, Informatics and Medical Engineering (PRIME), 2012 International Conference on*, pages 41–48, Salem, Tamilnadu, India, 21-23 Mar. 2012.

[5] E. Oksuzoglu and D. S. Wallach. Votebox nano: A smaller, stronger FPGA-based voting machine (short paper). In *Proceedings of the 2009 USENIX/Accurate Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, EVT/WOTE 09, Montreal, Canada, 10-14 Aug. 2009.

[6] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach. Analysis of an electronic voting system. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 27–40, Oakland, California, USA, 9-12 May 2004.

[7] N. Fauzia, T. Dey, I. Bhuiyan, and M. Rahman. An efficient implementation of electronic election system. In *Computer and information technology, 2007. ICCIT 2007. 10th international conference on*, pages 1–6, Gyeongju, Korea, 2007.

[8] M. Alam, M. Masum, M. Rahman, and A. Rahman. Design and implementation of microprocessor based electronic voting system. In *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on*, pages 264–269, Bangladesh, 2008.

[9] T. Wollinger, J. Guajardo, and C. Paar. Cryptography on FPGAs: State of the art implementations and attacks. In *ACM Transactions on Embedded Computer Systems, Vol 3, No 3, pp 534-574*, Aug. 2004.

[10] A. Lesea. IP security in FPGAs. In *Xilinx Inc.*, 2007.

[11] Y. M. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *16th USENIX Security Symposium (USENIX Security 2007)*, Boston, MA, USA, 6-10 Aug. 2007.

[12] S. Drimer and M. G. Kuhn. A protocol for secure remote updates of FPGA configurations. In *Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications*, ARC '09, pages 50–61, Berlin, Heidelberg, 2009.

[13] S. Dutt and L. Li. Trust-based design and check of FPGA circuits using two-level randomized ecc structures. *ACM Trans. Reconfigurable Technol. Syst.*, 2(1):6:1–6:36, Mar. 2009.

[14] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Des. Test*, 27:10–25, Jan. 2010.

[15] 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, Anaheim, CA, USA, 2008.

[16] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic. Hardware trojan detection and isolation using current integration and localized current analysis. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pages 87–95, Cambridge (MA), USA, 1-3 Oct. 2008.

[17] 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, Dec. 2010.

[18] M. Banga, M. Chandrasekar, L. Fang, and M. Hsiao. Guided test generation for isolation and detection of embedded trojans in ICs. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, GLSVLSI '08, pages 363–366, Orlando, FL, USA, 2008.

[19] M. Banga and M. S. Hsiao. A novel sustained vector technique for the detection of hardware trojans. In *Proceedings of the 22nd International Conference on VLSI Design*, pages 327–332, New Delhi, India, 5-9 Jan. 2009.

[20] M. Banga and M. S. Hsiao. VITAMIN: Voltage inversion technique to ascertain malicious insertions in ICs. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '09, pages 104–107, Washington, DC, USA, 2009.

[21] 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, 2009.

[22] 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.

[23] 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, June 2011.

[24] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Des. Test*, 27(1):66–75, Jan. 2010.

[25] A. Waksman and S. Sethumadhavan. Silencing hardware backdoors. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 49–63, Oakland, California, USA, 2011.

[26] M. Beaumont, B. Hopkins, and T. Newby. Safer path: Security architecture using fragmented execution and replication for protection against trojaned hardware. In *DATE*, pages 1000–1005, San Jose, CA, USA, 2012.

[27] A. Al-Anwar, Y. Alkabani, M. El-Kharashi, and H. Bedour. Defeating hardware spyware in third party IPs. In *Electronics, Communications and Photonics Conference (SIECPC), 2013 Saudi International*, pages 1–5, Saudi Arabia, 2013.

[28] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[29] A. Al-Anwar, Y. Alkabani, M. El-Kharashi, and H. Bedour. Hardware trojan protection for third party IPs on FPGA. In *16th EUROMICRO Conference on Digital System Design*, Santander Cantabria, Spain, 2013.