Multi-level Attacks: an Emerging Security Concern for Cryptographic Hardware

Sk. Subidh Ali*, Rajat Subhra Chakraborty*, Debdeep Mukhopadhyay* and Swarup Bhunia†
*Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, West Bengal–721302, India
Email: {subidh,rschakraborty,debdeep}@cse.iitkgp.ernet.in
†Department of Electrical Engineering and Computer Science
Case Western Reserve University, Cleveland, OH–44106, USA
Email: skb21@case.edu

***Abstract*—Modern hardware and software implementations of cryptographic algorithms are subject to multiple sophisticated attacks, such as differential power analysis (DPA) and fault-based attacks. In addition, modern integrated circuit (IC) design and manufacturing follows a horizontal business model where different third-party vendors provide hardware, software and manufacturing services, thus making it difficult to ensure the trustworthiness of the entire process. Such business practices make the designs vulnerable to hard-to-detect malicious modifications by an adversary, termed as "Hardware Trojans". In this paper, we show that malicious nexus between multiple parties at different stages of the design, manufacturing and deployment makes the attacks on cryptographic hardware more potent. We describe the general model of such an attack, which we refer to as *Multi-level Attack*, and provide an example of it on the hardware implementation of the *Advanced Encryption Standard* (AES) algorithm, where a hardware Trojan is embedded in the design. We then analytically show that the resultant attack poses a significantly stronger threat than that from a Trojan attack by a single adversary. We validate our theoretical analysis using power simulation results as well as hardware measurement and emulation on a FPGA platform.**

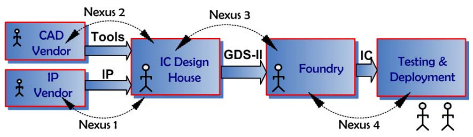
I. Introduction

To ease the computational burden of implementing complex cryptographic algorithms in security-sensitive applications, VLSI hardware ICs (crypto-chips) or highly optimized software routines (crypto-libraries) are commonly used. Normally, the crypto-algorithms are mathematically secure. However, based on the process of information leakage due to implementation issues, several sophisticated attacks [1]–[3] have been devised to break a crypto-system by discovering the secret key. Attacks based on analysis of the transient power of the crypto-circuit ("Differential Power Analysis") [1] or inducing faults in the circuit ("Fault Attacks") [2], [3] have been shown to be potent threats. Among these, the fault-attacks are particularly interesting since they require relatively less computational effort and are easy to launch. In recent years, the computational complexity of deducing the secret key by analysis of a single faulty cipher-text of the 128-bit version of the *Advanced Encryption Standard* (AES) cryptographic algorithm has been reduced to a brute-force search 2^{32} possibilities [3], which can be performed in a few minutes on standard modern desktops.

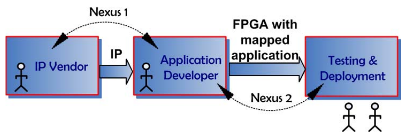
Side-by-side, economic reasons dictate the widespread outsourcing of design and manufacturing services to physically remote third-parties. Often, the design house procures pre-verified, ready-made components of their design from third-party vendors in the form of *hardware intellectual property* ("Hardware IP") modules. Besides, modern complex IC design is dependant largely on the availability of sophisticated computer-aided design (CAD) tools from software vendors. Finally, most semiconductor companies are now-a-days following a "fabless model" in which the design database is sent to remote fabrication facilities ("fabs") for manufacturing. All these reasons make a design vulnerable to malicious modifications, commonly referred to as *hardware Trojans* [6], [8]. These hardware Trojans are typically *stealthy* in nature such that they can easily evade conventional post-manufacturing testing. Once deployed in-field, they trigger and cause catastrophic system failure or leak secret information. Recent research has widely addressed the modeling and detection of these hardware Trojans [8].

Recent research has also focused on threats that result from specific nexus between parties associated with the design, manufacturing and deployment of cryptographic hardware. One such example is [4], where an inserted Trojan circuitry leaks information through a covert side-channel that allows conspiring malicious parties to discover the encryption key. Another example is [7], where with the help of *focused ion beams* an inserted dormant Trojan is connected to the functional unit in the fab. Such nexus poses serious threat to cryptographic applications, by combining established powerful attacks on cryptographic hardware, with malicious hardware Trojans that enable or facilitate these attacks for parties who are part of the nexus. Often, these hardware Trojans can be specifically enabled by only the malicious parties; hence, it becomes extremely difficult to detect them.

Although some specific instances of multi-level attack (MLA) have been explored in diverse contexts, in this paper



(a) Multi-level threat for an IP-based cryptographic IC design and manufacturing flow



(b) Multi-level threat for an IP-based FPGA design flow

Fig. 1. Example of multi-level attacks for both ASIC and FPGA realizations of cryptographic hardware.

we examine it in its general form as an issue arising out of modern design and manufacturing practices. As far as we know, this is the first work that presents the general model of multi-level attacks, gives an example of such an attack, analyses its effectiveness theoretically as well as through experimental validation. We show that multi-level attacks pose stronger threat in terms of evading existing defence mechanisms than the attacks involving a single party with access to a single level of IC life-cycle.

The rest of the paper is organized as follows. In Section II, we elucidate the concept of multi-level attacks with an example of an attack on a hardware implementation of AES. We present simulation and experimental validation results in Section III. We conclude in Section IV.

II. Multi-level Attacks: A Case Study

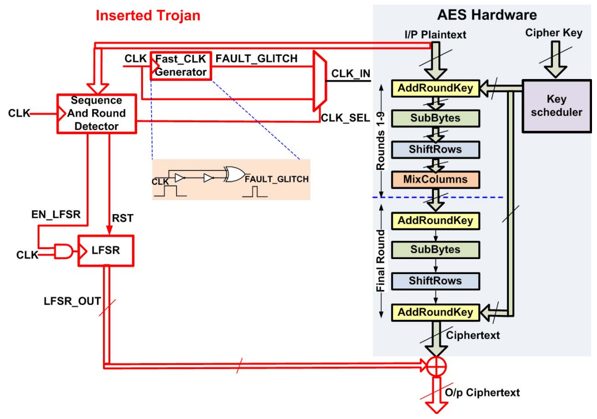
A. General Model

Fig. 1 shows possible nexus that might exist between different parties associated with the design, manufacturing and deployment in the life-cycle of IP-based cryptographic hardware. Fig. 1(a) and 1(b) consider the hardware life-cycle for Application Specific Integrated Circuit (ASIC) and field programmable gate array (FPGA) realization, respectively. Nexus between two or more stages can be leveraged to mount extremely strong attacks, leading to IP piracy, post-deployment malfunction or information leakage. Due to the distributed nature of these attacks, they can easily evade security verification at individual stages.

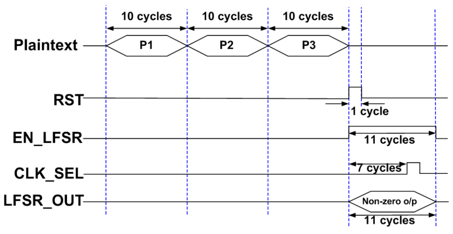
B. Attack on AES Hardware

Consider an iterative implementation of the Advanced Encryption Standard (AES) algorithm as shown in Fig. 2(a). A typical AES encryption consists of ten rounds of a series of operations - *AddRoundKey*, *SubBytes*, *ShiftRows* and *MixColumns*, with the tenth round replacing the *MixColumns* step with an extra *AddRoundKey* step. This particular implementation of AES takes ten clock cycles to complete the encryption of a single 128-bit block of plain-text.

Let us assume that the malicious adversary is the designer of the AES hardware herself, who wants to know the secret



(a) AES hardware with Trojan



(b) Timing diagram of the Trojan signals

Fig. 2. Example of a designer embedded Trojan in the hardware implementation of the AES cryptographic hardware.

encryption key after the hardware has been deployed. For this, the malicious designer develops a nexus with the person in charge of deploying the encryption hardware in-field. Typically, such encryption hardware is designed in the form of hardware intellectual property cores (IP cores) in hardware description language (HDL), which can be directly synthesized and either used as a building block in a cryptographic IC or mapped on a FPGA. The person in charge of deployment would allow an AES IP core infected with a Trojan to be included. The encryption key is usually hard-coded in a tamper-resistant non-volatile memory module on the circuit board, hence the deployer cannot directly access the key. Since multiple parties are involved here in trying to discover some secret information, this is an example of a *multi-level attack*.

The knowledge of how the hardware Trojan might be triggered is a secret shared only between the two co-operating malicious parties. The structure of the inserted Trojan circuit is shown in Fig. 2(a) and the associated timing diagram of the Trojan signals is shown in Fig. 2(b). The hardware Trojan is triggered (activated) by the application of three consecutive patterns (denoted by $P1$, $P2$, $P3$ in Fig. 2(b)) at chosen bit positions of the input plain-text, which are easily controllable by the in-field adversary. The probability of triggering such a Trojan accidentally by non-motivated application of plain-text is extremely small. To see this, consider that the Trojan is triggered by the occurrence of consecutively three different bit patterns each of length thirty at thirty different bit positions of the plain-text. Then, the probability of successfully applying one of the patterns is $\frac{1}{2^{30}}$. Since the choices of the patterns are independent of each other, the probability of successfully applying all the three patterns in correct sequence and triggering the Trojan is given by:

$$P_{trigger} = \left(\frac{1}{2^{30}}\right)^3 = \frac{1}{2^{90}} \approx 10^{-27} \qquad (1)$$

which is minuscule. Hence, we can safely assume that only the deployer can trigger the Trojan by a "chosen plain-text" attack, by controlling the plain-text input to the encrypter.

The Trojan activates after detecting this sequence of patterns through a sequence detector. The Trojan also includes a delay-based glitch generator circuitry which generates glitches ($FAULT\_GLITCH$) by XOR-ing the system clock with a delayed version of itself, as shown in Fig. 2(a). On activating, the Trojan waits for seven clock cycles before enabling a multiplexor through the CLK_SEL signal that lets a narrow glitch ($FAULT\_GLITCH$) being applied at the clock input instead of the system-clock. This causes a *setup time violation* in the input flip-flops when the seventh round cipher-text is fed-back to the input of the encryption hardware to start the eighth round encryption [5]. Thus, the inserted Trojan hardware injects a *fault* in the AES circuit at the beginning of the eighth round during encryption. It has been shown [3] that by analyzing two faulty cipher-texts corresponding to two known plain-texts, the 128-bit AES key can be deduced exactly without any brute-force search. If only one plain-text cipher-text pair is known, the key can be deduced exactly by a brute-force search of the order of 2^{32}.

The adversaries want that *only they and no other party* would be able to deduce the secret key from the faulty cipher-text. To achieve this, she can mask the cipher-text by adding a *Linear Feedback Shift Register* (LFSR) to the design. The LFSR remains active for the time taken to encrypt a single plain-text by the AES hardware. The state transitions of the LFSR is controlled by the EN_LFSR signal which is synchronized with the event of multiple successful pattern matches. This concept takes its motivation from the type of information leakage Trojan described in [4]. The infeasibility of recovering the key with a modified faulty cipher-text is shown in Sec. II-C.

C. Difficulty of Recovering Encryption Key from Modified Faulty Cipher-text

Let us assume that the output cipher byte c_i is being masked by XOR-ing with eight selected output bits of the $LFSR$. Therefore c_i must be part of one of the four ninth round system of equation proposed in paper [3]. Let us assume the corresponding quartet of key bytes are $\{k_p, k_q, k_r, k_s\}$ and the corresponding system of equations is:

$$\begin{aligned} 2\delta &= S^{-1}(x_1 \oplus k_p) \oplus S^{-1}(x'_1 \oplus k_p) \\ \delta &= S^{-1}(x_2 \oplus k_q) \oplus S^{-1}(x'_2 \oplus k_q) \\ \delta &= S^{-1}(x_3 \oplus k_r) \oplus S^{-1}(x'_3 \oplus k_r) \\ 3\delta &= S^{-1}(x_4 \oplus k_s) \oplus S^{-1}(x'_4 \oplus k_s) \end{aligned}$$

where x_1, x_2, x_3, x_4 are the actual cipher-text, and x'_1, x'_2, x'_3, x'_4 are the corresponding faulty cipher-text values. Here $\delta \in \{0, \ldots, 255\}$ and S^{-1} represents the *InverseSubByte* operation of AES. We now prove by contradiction that the masked output cipher-text will not reveal the secret key.

Let us assume that the masked cipher reveals the actual key and x_1 in above equation represents c_i and the corresponding faulty cipher byte and the masked faulty cipher bytes are x'_1 and $x'_1 \oplus \alpha$, where α is a non-zero masked value generated by the LFSR. Therefore, the above equation should give same quartet of key bytes $\{k_p, k_q, k_r, k_s\}$ with the masked value. In that case only the first equation changes and the rest of the equations remain unchanged. Hence, from the first equation we can write,

$$\begin{aligned} &S^{-1}(x_1 \oplus k_p) \oplus S^{-1}(x'_1 \oplus \alpha \oplus k_p) \\ &\quad = S^{-1}(x_1 \oplus k_p) \oplus S^{-1}(x'_1 \oplus k_p) \end{aligned}$$

which implies

$$S^{-1}(x'_1 \oplus \alpha \oplus k_p) = S^{-1}(x'_1 \oplus k_p)$$

which implies $x'_1 = x'_1 \oplus \alpha$ and $\alpha = 0$, since the S^{-1} mapping is bijective. This conclusion contradicts our assumption.

D. Effectiveness of Multi-level Attacks

We now try to quantitatively estimate the effectiveness of an attack based on the nexus between multiple parties at different levels. Consider the example given above. In this case, note that it is not fruitful to consider whether the nexus between the designer and the deployer makes it easier for either party to launch an attack. It would be generally infeasible (even for the deployer) to launch a fault-attack on the AES encrypter by suddenly increasing the clock frequency to the encrypter. The effectiveness of the attack can be realized by estimating the difficulty in discovering the inserted hardware Trojan, and then using it to retrieve the encryption key, for somebody who is not part of the nexus. To discover the scheme, a third-party (who is not part of the nexus) must perform two tasks successfully:

- Activate the inserted Trojan by applying the three correct patterns ($P1$, $P2$ and $P3$), and,
- Identify the bit positions of the output cipher-text whose values have been inverted by the Trojan LFSR.

In general, if each of the Trojan activation sequence vectors is M-bit long and the length of the initialization sequence is N, the complexity of activating the Trojan by a brute-force method is $O(2^{M \cdot N})$. To perform the second task successfully by brute force (since a third-party has no way of knowing this information), P bit positions out of 128 bits AES cipher-text must be chosen, and corresponding to each of the assumed choices, an average of 2^{32} operations must be performed to calculate the key. Only one of these operations of overall complexity $O\left(\binom{128}{P} \cdot 2^{32}\right)$ will yield the correct key. Hence, for a third-party to actually launch a successful fault-attack on the above hardware will require brute-force operations of complexity $O\left(2^{M \cdot N}\binom{128}{P} \cdot 2^{32}\right)$. For example, with $M = 10$, $N = 3$, $P = 15$ (i.e. fifteen bits of the output cipher-text were flipped), the above complexity is $\approx 2^{126}$,

TABLE I
SIMULATED INCREASE IN AVERAGE POWER

Design	Average Power (mW)	% Increase (w.r.t. golden)
AES without Trojan (golden)	120.60	0.00
AES with Trojan (LFSR inactive)	120.68	0.07
AES with Trojan (LFSR active)	120.86	0.22

TABLE II
HARDWARE OVERHEAD

Design	Slices	Slice Flip-flops	4-input LUTs
AES without Trojan (golden)	3229	2437	5835
AES with Trojan	3260	2487	5898
Overhead (%)	0.96	2.05	1.08

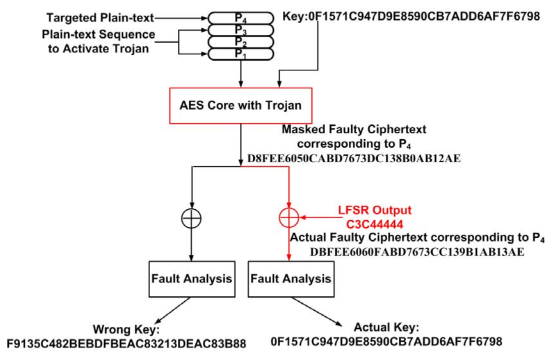

Targeted Plain-text
Plain-text Sequence to Activate Trojan
P_4
P_3
P_2
P_1
Key:0F1571C947D9E8590CB7ADD6AF7F6798
AES Core with Trojan
Masked Faulty Ciphertext corresponding to P_4
D8FEE6050CABD7673DC138B0AB12AE
LFSR Output
C3C44444
Actual Faulty Ciphertext corresponding to P_4
DBFEE6060FABD7673CC139B1AB13AE
Fault Analysis
Fault Analysis
Wrong Key:
F9135C482BEBDFBEAC83213DEAC83B88
Actual Key:
0F1571C947D9E8590CB7ADD6AF7F6798

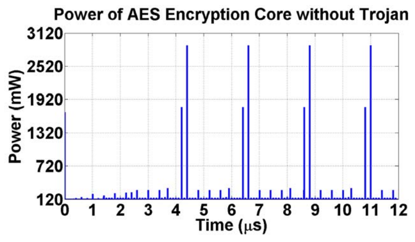
Fig. 4. Experimental setup to simulate multi-level attack.

which is comparable to the complexity of finding an 128-bit AES encryption key by a brute-force search.

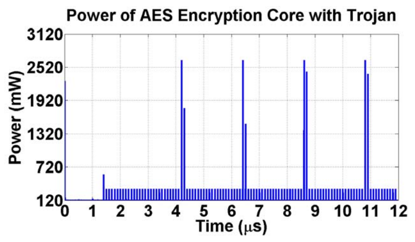
III. RESULTS

The iterative AES encryption core infected with a length-3, 10-bit pattern detector and a 32-bit LFSR-based hardware Trojan as described in Section II was implemented in Verilog and simulated using *ModelSim*. The design was synthesized using *Xilinx ISE* to map it to a *Xilinx Spartan-3E* FPGA board. *Xilinx XPower* was used to simulate the transient power trace of the circuit. Fig. 3 shows the simulated power traces of the circuit with and without Trojan. Table I shows the percentage increase in the average power consumption of the infected design as compared to the golden design, and Table II shows the hardware overhead. As is evident from these two tables, the Trojan is small relative to the original circuit and has negligible effect on the average power consumption, and is thus extremely difficult to detect using side-channel techniques which are commonly affected by experimental noise and process variation effects.

To show the effectiveness of the above multi-level attack scenario, a fault attack was launched using a glitch as shown in Fig. 4. When the effect of the masking by the output of the LFSR was not considered, the fault analysis technique described in [3] yielded an incorrect key, which was different in all the sixteen bytes compared to the original key. When the effect of the masking was taken into consideration, the correct key was recovered through the fault analysis attack, as expected.



(a) Power trace of circuit without Trojan



(b) Power trace of circuit with Trojan

Fig. 3. Power traces of circuits with and without Trojan.

IV. CONCLUSIONS

In this paper we have analyzed and exemplified a new class of hardware attacks on cryptographic algorithms arising out of the nexus between different parties. This attack utilizes the nexus to preferentially launch a *fault-attack*, and enables only the malicious parties, who are part of the nexus to retrieve the cipher key. We have shown that such an attack can be realized using minimal hardware overhead, and has negligible effect on the power consumption, which makes its detection extremely challenging. We have demonstrated the greater potency of such multi-level attack compared to single-level attacks through analysis, simulations and FPGA emulations.

ACKNOWLEDGEMENT

The authors would like to thank Bodhisatwa Mazumdar for his help in analyzing the FPGA power traces.

REFERENCES

[1] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis", *Lecture Notes on Computer Science*, vol. 1666, pp. 388–397, 1999.
[2] F. Amiel, C. Clavier and M. Tunstall, "Fault analysis of DPA-resistant algorithms", *Lecture Notes on Computer Science*, vol. 4236, pp. 223–236, 2006.
[3] D. Mukhopadhyay, "An improved fault based attack of the Advanced Encryption Standard", *Lecture Notes on Computer Science*, vol. 5580, pp. 421–434, 2009.
[4] L. Lin, W. Burleson and C. Parr, "MOLES: malicious off-chip leakage enabled by side-channels", *Proceedings of the International Conference on CAD*, pp. 117–122, 2009.
[5] N. Salmane, S. Guilley and J. Danger, "Practical setup time violation attacks on AES", *Proceedings of the European Dependable Computing Conference*, pp. 91–96, 2008.
[6] DARPA, "TRUST in Integrated Circuits (TIC)". 2007. [Online]. Available: http://www.darpa.mil/MTO/solicitations/baa07-24.
[7] Miron Abramovici, "Protecting integrated circuits from silicon Trojan horses", *Military Embedded Systems*, Jan–Feb, 2009. [Online]. Available: http://www.mil-embedded.com/articles/id/?3748.
[8] R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: threats and emerging solutions", *Proceedings of the IEEE International High Level Design Validation and Test Workshop*, pp. 166–171, 2009.