Hardware Dithering: A Run-Time Method for Trojan Neutralization in Wireless Cryptographic ICs

Christiana Kapatsori, Yu Liu, Angelos Antonopoulos and Yiorgos Makris
Department of Electrical and Computer Engineering
The University of Texas at Dallas
Richardson, TX 75080
Email: {christiana.kapatsori, yxl119120, aanton, yiorgos.makris}@utdallas.edu

***Abstract*—We introduce a hardware dithering methodology for neutralizing Trojans in integrated circuits (ICs). The proposed approach seeks to make the operating point of an IC an unpredictable moving target during run time. Thereby, the ability of a Trojan to exploit the process variation margins, wherein hardware Trojans typically find breathing room to operate while remaining concealed, is significantly restricted. To demonstrate this hardware dithering concept, we leverage tuning knobs operating on the power and frequency characteristics of the transmission of a wireless cryptographic IC. These knobs are driven by a random number generator, thus forcing the circuit into a random walk in the space of its parametric performances while in normal operating mode. In essence, while the circuit remains within its operating specifications during this random walk, its exact operating point varies, thus muddying the waters for the adversary. Experimental results on the wireless cryptographic IC, which was designed and fabricated in a 0.35μm CMOS technology, corroborate that hardware dithering imposes a significant and unpredictably dispersed bit error rate to the adversary, thereby impeding hardware Trojan operation.**

I. Introduction

Hardware Trojan concepts described in the extensive relevant literature of the last decade [2]–[4], [18] are generally driven by two fundamental objectives: (i) they seek to introduce computational errors in the operation of a chip or even incapacitate it, or (ii) they seek to leak information through covert side channels, such as timing, power, electromagnetic emanations, etc. While both of these hardware Trojan classes pose a serious threat to the trusted and secure operation of contemporary electronics, the silent nature of the latter, which incurs no evident interference with correct functionality of a circuit, makes it much more challenging to defend against.

As a result, numerous methods have been proposed for detecting hardware Trojans which establish and/or exploit covert side channels [1], [3], [7]. The vast majority of these methods rely on a statistical approach known as *side-channel fingerprinting*. In this approach, a parametric signature, typically consisting of continuous-domain measurements such as power, delay, temperature, electromagnetic interference, or combinations thereof, is obtained for a suspect chip and statistically compared against a distribution of signatures originating from known Trojan-free (golden) chips. Advanced statistical side-channel fingerprinting methods have pushed the envelope via sophisticated machine learning-based solutions for increasing classification accuracy [9], [15], reducing or eliminating reliance on golden chips [7], [10], and even providing real-time hardware Trojan monitoring through on-die classifiers [12].

Nevertheless, statistical fingerprinting remains a *reactive* approach, which can only detect Trojans that leak information through covert side channels once such leakage has already taken place. In many cases, such detection may already be too late, as even a very small bandwidth may suffice to compromise security and privacy by leaking cryptographic keys, credit card numbers, personal identification numbers, or other sensitive data. Therefore, *proactive* methods which seek to prevent either insertion or robust operation of hardware Trojans are also required to ensure security and trust.

To this end, in this paper we introduce *hardware dithering*, a prevention method which seeks to challenge the operational underpinnings of hardware Trojans that utilize covert side channels and, thereby, to neutralize them. Specifically, the key contributions of this work include:

- Introduction of hardware dithering as a hardware Trojan neutralization method and description of its underlying philosophy for preventing robust Trojan operation.
- Detailed implementation of hardware dithering in the context of a wireless cryptographic transmitter.
- Experimental impact and effectiveness characterization of hardware dithering using over-the-air measurements from fabricated Trojan-free and Trojan-infested versions of the aforementioned wireless cryptographic transmitter.

We note that we chose the wireless cryptographic IC domain as a platform for demonstrating hardware dithering for two reasons: (i) encrypted sensitive and confidential information is typically exchanged in this domain, making it an *attractive* attack target, and (ii) such exchange takes place over public communication channels, making it a *plausible* attack target, without requiring physical access to the IC. However, the operating principles of hardware dithering are independent of this domain and can be generally applied to protect any circuit from covert side-channel Trojans.

II. Side-Channel Trojan Operation

The fundamental enabler of covert side-channel hardware Trojans is the inherent *process variation* of semiconductor manufacturing. Indeed, the acceptable margins allowed in the performances of a chip population, in order to compensate for the inability to precisely control the IC fabrication process, provide breathing space for such Trojans.

978-1-5386-8382-8/18/$31.00

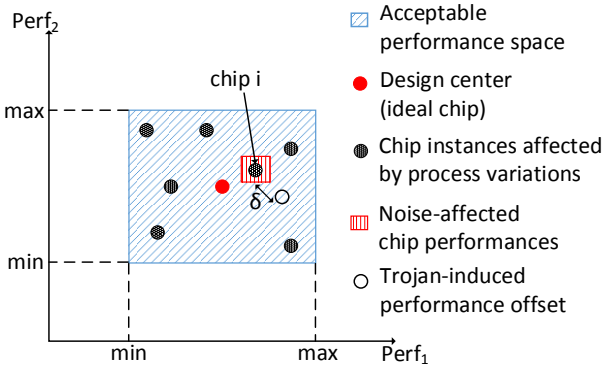


Fig. 1: Operating principle of covert side-channel Trojans.

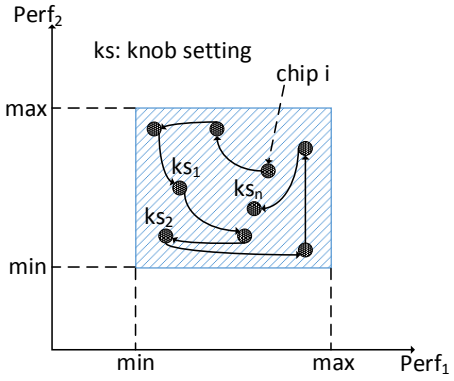


Fig. 2: General concept of hardware dithering.

Consider, for example, a simple design with two continuous-domain performances, $Perf_1$ and $Perf_2$, as shown in Fig. 1. While the targeted design center is a single point in the space of these two performances, fabricated chips will exhibit a distribution of performances around this design center due to process variation. Therefore, minimum and maximum values are defined for each performance in order to specify the region wherein acceptable chips are expected to operate. Within this space, operation of each fabricated chip is represented by a single point, depending on how it has been affected by process variation. However, to account for operational fluctuations and measurement noise, a small window around each such point is used to reflect its operating region.

Let us now consider a hardware Trojan which seeks to establish a covert side channel using these two performances, in order to leak information. All the hardware Trojan needs to do is to systematically induce an offset in the operating point of a circuit. Indeed, consider chip instance i in our example, which in addition to its original operating point, can also be controlled by a Trojan to operate at an offset point. As long as the offset δ exceeds normal operating fluctuations and measurement noise, the Trojan has successfully established a covert communication channel. Specifically, by encoding logic '0' on the original operating point and logic '1' on the offset operating point (or vice versa), the Trojan can now leak data through the two performances of the chip, by systematically jumping between its two operating points.

Inconspicuousness of such Trojans stems from the fact that both operating points are within the acceptable performance space established for dealing with process variations. Indeed, in the eyes of the unsuspecting user, the minute fluctuation between the two operating points is perceived as increased operational or measurement noise and will not trigger an alarm, as the chip remains within its acceptable performance space. In the eyes of the knowledgeable attacker, however, this fluctuation is sufficient for retrieving the leaked data.

We emphasize that information leakage through such covert channels is based on the *offset* between the two operating points, not their actual values. Indeed, as the hardware Trojan is introduced prior to fabrication, the exact operating point of each chip is unknown. Therefore, the attacker may only rely on the offset induced by the Trojan. We also note that, in this example, we only used one offset operating point, as this is the minimum required for differentiating between the two possible values of a leaked bit. Additional offset points may potentially be used to increase the density of information encoded on the circuit operating point.

III. Hardware Dithering

While the simple motivational example of the previous section highlights the operating principle of hardware Trojans which establish covert side channels, it also points at a possible solution for preventing their robust operation. More specifically, it reveals the *need for a stable operating point* for a chip, subject only to small operating fluctuations and measurement noise. This is an indispensable requirement, as the offset from this stable point is the mechanism through which such Trojans can encode and leak information.

Accordingly, the proposed hardware dithering method seeks to challenge availability of such a stable operating point. In essence, as depicted in Fig. 2, hardware dithering sends the circuit on a random walk within the space of its performances. During this walk, the chip remains within its design specifications, yet its exact operating point changes continuously. Thereby, the stable reference operating point becomes a moving target and any offsets computed during this walk are distorted by a random component. Hence, their ability to robustly encode and leak information is diminished.

To implement hardware dithering, various tuning knobs must be added to the circuit, in order to modulate its performances. During normal operation, these knobs are driven by random value sequences, forcing the circuit into the aforementioned random walk. While knob implementation and random value generation details may vary, effective hardware dithering relies on the following constraints:

- Legitimate operation of the circuit should remain unaffected by hardware dithering.
- Selection and design of tuning knobs must be Trojan-agnostic, i.e., it should have no knowledge of and make no assumptions about the operation of Trojans, except that they seek to leak data through circuit performances.
- Tuning knobs should be an inherent part of the design, so that an attacker may not simply identify and remove them

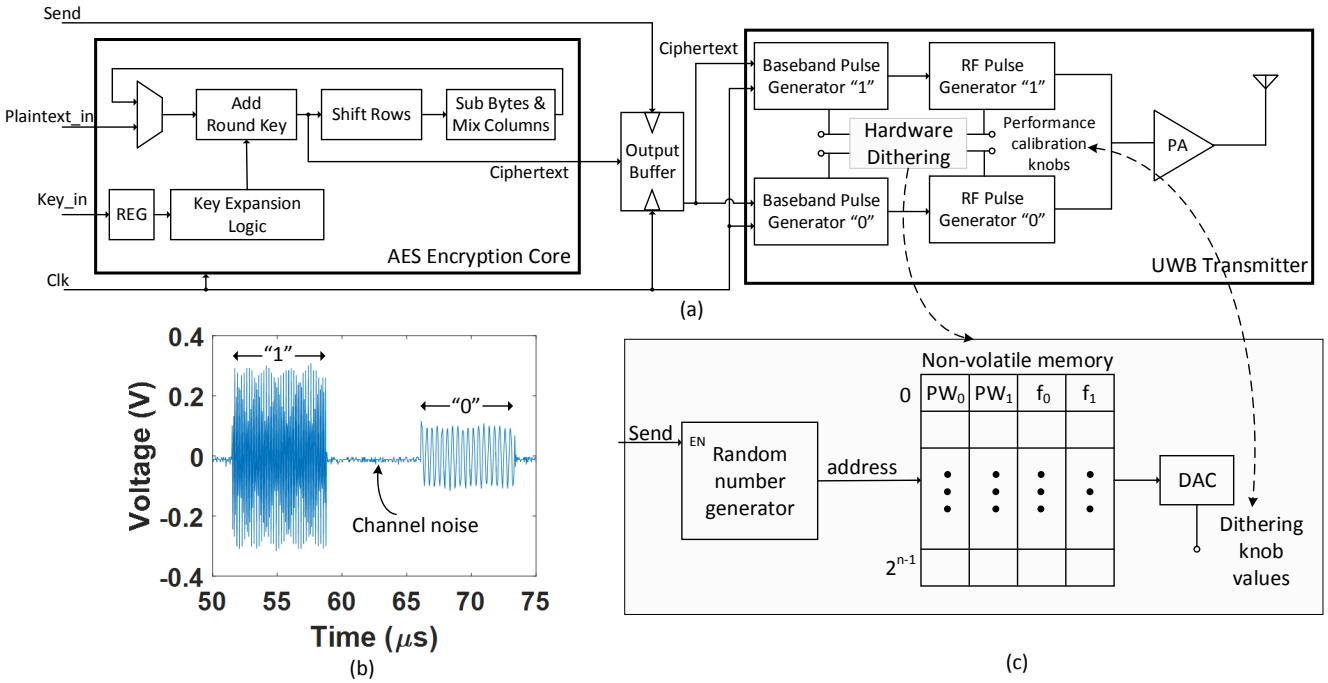


Fig. 3: (a) System-level block diagram of the wireless cryptographic IC, (b) transmission voltage while sending "1" and "0", and (c) dithering implementation.

during Trojan insertion. In other words, correct operation should rely on some value provided through these knobs.

- The attacker should not be privy to the random sequence of values used to drive the tuning knobs. Therefore, these values should be programmed in each chip after it is fabricated, through the use of non-volatile or other one-time-programmable memory.
- The attacker's ability to extract the leaked information from covert side-channel measurements in the presence of hardware dithering should be significantly hindered.

IV. Experimental Platform

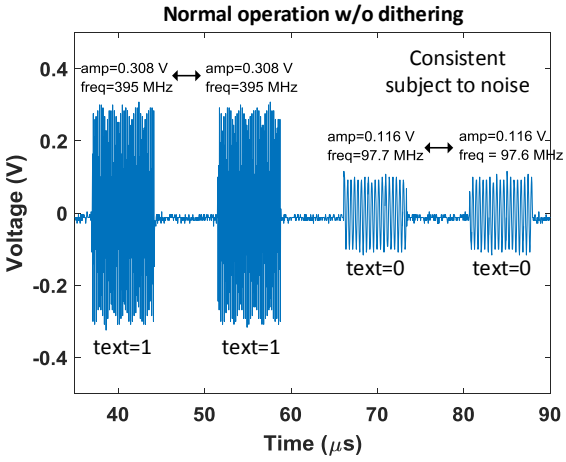
To demonstrate hardware dithering we use as a starting point the wireless cryptographic benchmark design developed in [11]. This design consists of a digital and an analog part implementing an advanced encryption standard (AES) core and an ultra wide-band (UWB) transmitter, respectively. Below, we first review its normal operation, and then describe the modifications required for implementing hardware dithering.

Normal Operation: A system-level block diagram of the design is shown in Fig. 3(a). The AES core receives plaintext in blocks of 128 bits, which it encrypts using a 128-bit key that is loaded through the `key_in` input and stored on-chip [11]. After ten rounds of transformation, the plaintext is encrypted into ciphertext, which is stored in an output buffer in blocks of 128 bits, until it is transmitted. The output buffer serializes the ciphertext before sending it to the UWB transmitter. The UWB transmitter comprises a baseband pulse generator, an RF pulse generator and a power amplifier (PA). As shown in Fig. 3(b), transmission of a logic 1 signal has higher amplitude and higher frequency than transmission of a logic 0.

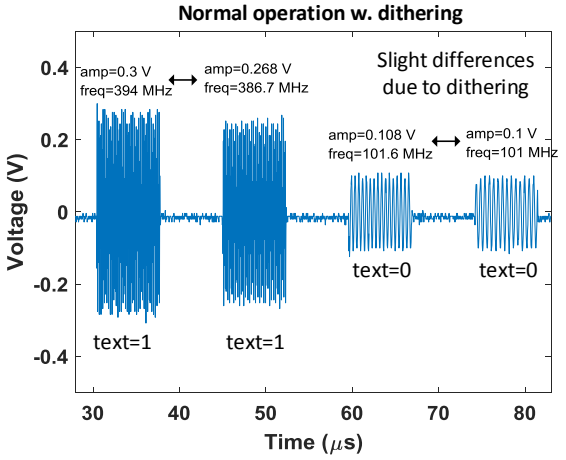
Performance Calibration Knobs: Most contemporary high-performance analog/RF IC designs are inherently equipped with performance calibration knobs which are tuned after manufacturing in order to ensure high yield in the presence of process variations [13], [14], [16]. The exact values of these tuning knobs are individually chosen for every IC and are either physically adjusted (e.g., laser resistor trimming) or, more commonly, permanently stored through non-volatile or one-time-programmable memory.

In the case of our wireless cryptographic IC, four tuning knobs exist as an inherent part of the design, in order to adjust the performances of each chip to its specifications after production. These four knobs, namely PW_0, f_0, PW_1, and f_1, come in the form of bias voltages and act on the amplitude and frequency characteristics of the wireless transmission. Two of them (PW_0, f_0) control the "0s" and two (PW_1, f_1) control the "1s". PW_i operates on the baseband pulse generator of the UWB transmitter and controls the gate terminals of nMOS transistors. With higher PW_i values, the current that flows through the corresponding branches is increased; thereby, so does the amplitude of the transmitted signal. On the other hand, f_i mainly determines the frequency of the RF pulse generator, again by controlling branch currents.

Inherent Knob Utilization for Hardware Dithering: To implement hardware dithering, we exploit the presence of the above four tuning knobs to drive the cryptographic transmitter into an unpredictable, random walk in the space of the transmission performances (i.e., amplitude and frequency), in order to reduce the robustness of any covert side-channel. To this end, we implemented the system depicted in Fig. 3(c),



(a)



(b)

Fig. 4: Dithering effect on normal IC operation.

which consists of: (i) a random number generator, and (ii) a non-volatile memory (NVM). The random number generator [5], [17] produces a random sequence wherein each value is used as an address to access the NVM. Each NVM entry contains a code (i.e., a vector of four digital values) for the four knobs. These knob codes are then provided to a digital-to-analog converter (DAC) which produces the analog voltages that determine the biases for knobs PW_0, PW_1, f_0, and f_1. Transmission amplitude and frequency are adjusted per bit. Therefore, among the knob values stored in the NVM, a different setting is applied to every bit.

We emphasize that the knobs and the DAC are an inherent part of the original cryptographic IC [11], with programmable constant values enabling adjustment of its operation across different specification requirements. Thus, they cannot be removed by an adversary, as the circuit will not be operable without them. Hence, only the random number generator and the NVM are exclusively added for the purpose of dithering.

We also note that the use of an NVM rather than a ROM is preferred for two reasons. First, attackers implanting HTs should not have access to the code values stored in the NVM, as they could potentially use this knowledge to evade the proposed scheme. Therefore, the codes need to be programmed

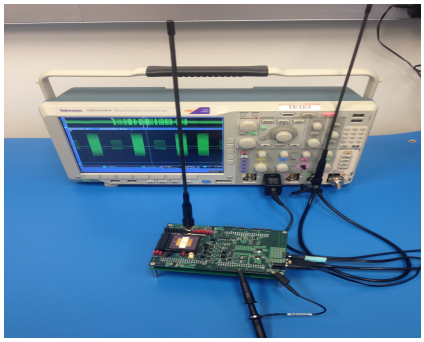


Fig. 5: Experimentation platform.

in the NVMs after the chips are manufactured. Second, due to manufacturing process variations, these values may slightly vary across chips, in order to ensure that, for each code, the circuit remains within its acceptable operating region.

The acceptable range of voltage values for each of the knobs and, by extension, the set of acceptable codes, is defined by the specifications of the UWB transmitter and can be determined prior to fabrication through Monte Carlo simulations. Among the many possible values in this set, the entries of the NVM for a specific chip are eventually selected based on two criteria: (a) for any chosen code, transmission of ciphertext bits of value "1" should remain robustly distinguishable from ciphertext bits of value "0", and (b) the subset of chosen codes should result in a uniform distribution of transmission characteristics (i.e., amplitude and frequency) in their acceptable space.

As demonstrated in Fig. 4, where four ciphertexts of "1" and "0" are plotted versus time before (top) and after (bottom) dithering is applied, normal operation of the IC is not impacted. Depending on the knob value, dithering may either increase or decrease transmission amplitude and frequency for both ciphertexts. However, it does not affect the ability of a receiver to correctly receive data bits, since ciphertexts "1" and "0" always remain distinguishable.

The described wireless cryptographic design, including the proposed hardware dithering capabilities, has been fabricated in TSMC's 0.35μm CMOS process. As shown in Fig. 5, our experimentation platform consists of a custom-designed PCB, which houses the wireless cryptographic chip with the hardware dithering knobs, as well as an oscilloscope. An antenna is connected to the transmitter's output in order to enable wireless communication. Another antenna connects to the oscilloscope, which acts as a receiver.

V. Threat Model & Sample Trojan

A. Threat Model

The general threat model considered in this work is depicted in Fig. 6 and consists of three entities, namely Bob, Alice and Eve, whose role is described below:

- **Alice** is a legitimate user who transmits information over a public wireless channel. Alice's transmitter has been contaminated by an adversary, either in the design or in the fabrication stage. The hardware Trojan which has been embedded in the wireless device creates a covert side-channel which aims to leak additional information

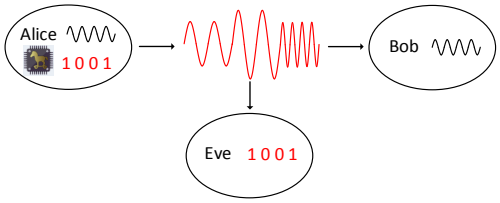


Fig. 6: Threat model.

by systematically modulating the parameters of the publicly accessible transmission. For example, Alice may be transmitting ciphertext and the hardware Trojan may be leaking the encryption key. Alice is not aware that a malicious operation is taking place.

- **Bob** is the legitimate entity that Alice is communicating with. As such, he is able to receive the data transmitted over the public channel but he is oblivious to the fact that additional information has been transmitted by Alice. In our example, where Alice is transmitting ciphertext, Bob is unaware that the key is also leaked through a covert side-channel of the transmission. In fact, Bob already possesses the key and is able to decipher the transmission.
- **Eve** is the malicious entity (rogue receiver) who seeks to receive the information leaked by the hardware Trojan in Alice's transmitter. Eve is not necessarily the person who embedded the hardware Trojan, yet Eve is collaborating with that person. As such, Eve is privy not only to the Trojan's presence in Alice's transmitter but also to the exact mechanism through which it establishes a covert side-channel through the publicly available wireless transmission. In fact, Eve is sufficiently intelligent to effectively retrieve the information leaked by the Trojan despite the noisy wireless channel conditions and the unique way in which process variations have affected Alice's transmitter during fabrication. Eve achieves this by using self-referencing, i.e., by relying on offsets rather than absolute values, as well as by employing statistical information retrieved from previously received data. In our example, Eve knows that the encryption key for Alice's ciphertext is leaked by modulating the wireless transmission parameters (e.g., amplitude, frequency) in a specific way and can, thus, retrieve it and compromise the encrypted communication between Alice and Bob.

B. Sample Trojan

To evaluate effectiveness of hardware dithering, we contaminated our wireless cryptographic IC with a sample hardware Trojan, which we designed and implemented based on the principles of the attacks introduced in [11]. This Trojan leaks the AES key by reading it from the register wherein it is stored and passing it to the transmitter. Therein, a handful of added transistors use the key bit values to modulate transmission amplitude and frequency. Specifically, regardless of the ciphertext value being transmitted, when the key bit being leaked is equal to "1" the Trojan increases slightly the amplitude and frequency coefficients of the transmission, creating the

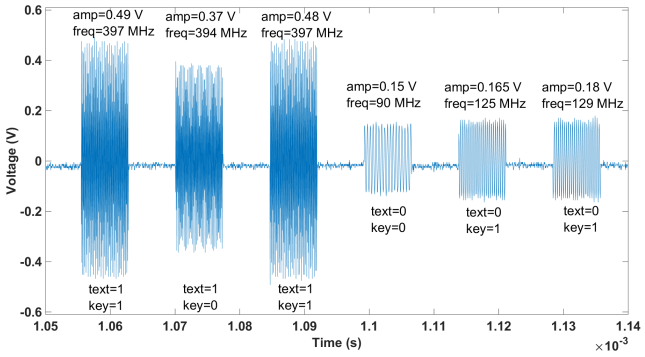


Fig. 7: Trojan operation.

offset required for establishing a covert channel, as discussed in Section II. Operation of the Trojan is depicted for six bits in Fig. 7, with measured amplitude and frequency annotated on each bit.

In order to retrieve the leaked key bits, a rogue receiver relies on the knowledge that the Trojan has introduced some positive offset in the 2-dimensional amplitude and frequency space. This offset is depicted in Fig. 8, where we plot the normalized frequency and amplitude levels for one transmission consisting of 128 bits[1]. As may be observed, the Trojan impact is more prominent on amplitude when the ciphertext is equal to "1" and on frequency when the ciphertext is equal to "0". The rogue receiver collects all these values and statistically analyzes them to retrieve the leaked key. In our case, a simple k-means clustering algorithm with $k = 4$ suffices for perfectly interpreting both the ciphertext and the key values. We reiterate that the Trojan does not interfere with the legitimate receiver's ability to correctly interpret the ciphertext values. Indeed, as shown in Fig. 8, the small offset introduced by the Trojan leaves plenty of separation between the ciphertext "1" (top/right) and ciphertext "0" (lower/left) measurements. Therefore, the simple thresholds typically employed by the unsuspecting legitimate receiver will perfectly classify these two populations.

VI. Experimental Results

Using the experimental platform described in Section IV, we evaluate the effectiveness of hardware dithering in preventing information leakage by the sample Trojan described in Section V-B. Specifically, we quantify (i) the number of errors that the rogue receiver experiences due to hardware dithering when retrieving the leaked key, and (ii) the distribution of errors across the bit locations of the transmitted 128-bit packets. *We note that, theoretically, maximal effectiveness is achieved when 50% of the transmitted bits are received erroneously and the probability of an error occurring at a specific location of the 128-bit packet is 0.5.*

We started by repeating the transmission of the same 128-bit ciphertext (encrypted with the same 128-bit AES key) as in the experiment depicted in Fig. 8, only this time with hardware

[1]We note that many of these 128 bits have identical amplitude and frequency values and, thus, overlap in the plot.

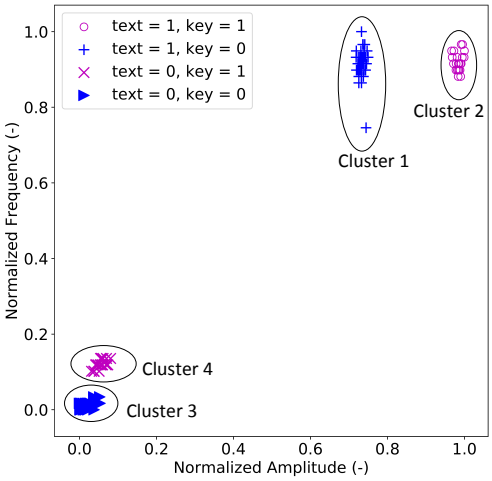


Fig. 8: Retrieving the leaked key.

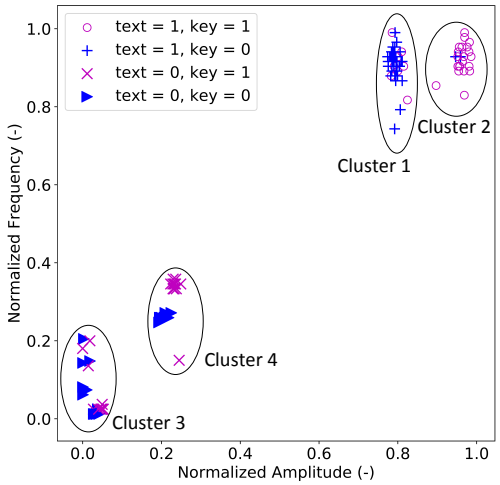


Fig. 9: Dithering effect on Trojan operation.

dithering enabled. The resulting amplitude and frequency measurements obtained by the rogue receiver are plotted in Fig. 9. As may be observed, by forcing the transmitter into a random walk wherein amplitude and frequency are uniquely perturbed for each ciphertext bit transmission, hardware dithering significantly hinders the ability of the rogue receiver to correctly retrieve the leaked key. While four distinguishable clusters are still observed, several transmitted bits have moved between cluster 1 and cluster 2, as well as between cluster 3 and cluster 4. Accordingly, for all such cases, the rogue receiver misinterprets the leaked key bit. Notably, despite hardware dithering, a large space remains between the ciphertext "1" (top/right) and ciphertext "0" (lower/left) populations; hence the ability of the legitimate receiver to distinguish them remains unaffected.

We then transmitted 100 randomly chosen 128-bit packets and calculated the number of bit errors occurring per packet at the rogue receiver after applying the clustering approach for retrieving the key bits. As shown in Fig. 10, the number of bit errors per packet ranges between 40 and 80 out of 128, whereas at the same time the legitimate user does not experience any errors. On average, the bit error rate per

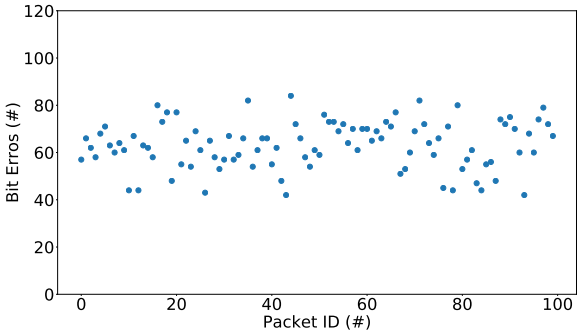


Fig. 10: Number of bit errors for 100 over-the-air transmissions of 128 bits each.

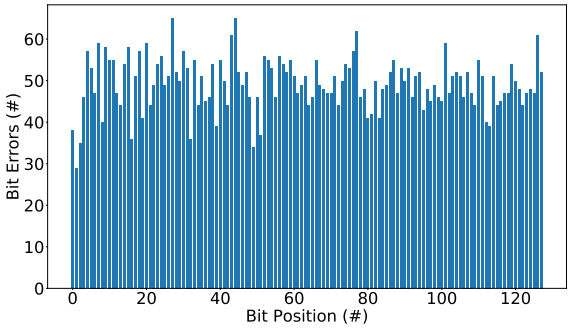


Fig. 11: Number of bit errors in each of the 128 bit positions for 100 over-the-air transmissions.

transmission is calculated at 49.3%, which is very close to the ideal theoretical value of 50%.

Finally, in Fig. 11, we show the number of errors that occurred in each bit position of the transmitted 128-bit packets, over the same 100 over-the-air experiments. On average, each of the 128 positions experienced 50 errors among the 100 transmissions, leading to a probability of exactly 0.5. Combined with the aforementioned bit error rate of 49.3%, this result confirms that hardware dithering has minimized the information that may be retrieved by the rogue receiver and has, effectively, neutralized the hardware Trojan.

VII. Related Work

The limited hardware Trojan prevention literature follows two directions. In the first direction, solutions seek to occupy empty space and unused resources in order to deprive attackers from the ability to hide a Trojan. For example, BISA [19] fills all empty space in a layout with functional standard cells, implementing additional functionality independent of the original design. Similarly, the FPGA-based protection scheme presented in [8] fills unused FPGA resources with low-level dummy logic, so that no free configurable resources exist for inserting a hardware Trojan in the design bit-stream. While these methods raise the difficulty level, an experienced attacker may still be able to identify the superfluous structures in the layout file or the FPGA bit-stream and remove them to make room for a hardware Trojan, or may introduce the Trojan at earlier design stages (e.g., through 3rd-party IP). Furthermore, such structures may significantly increase static

power consumption or even interfere with IC performances due to parasitics, especially in analog/RF ICs. In the second direction, which is closer to the hardware dithering approach, noise injection is used to thwart Trojan operation. For example, the attenuated signature noise injection method proposed in [6] uses a linear dropout regulator (LDO) to desensitize supply current from the encryption operation and, thereby, suppress data leakage through AES current traces. Besides defending only against power side-channels, the key limitation of this method is that the LDO is added exclusively for this purpose, rather than being an indispensable part of the AES design. Therefore, an adversary may remove it in the fabrication stage.

In contrast, hardware dithering: (i) is effective against intelligent hardware Trojans which create covert side-channels to leak information without violating any IC specifications, (ii) is Trojan-agnostic and can defend against attacks aiming to exploit process variation margins independent of the stage they are introduced in, and (iii) is based on tuning knobs which are integral to the design and cannot be removed, as this would result in a non-functional circuit.

VIII. Conclusion

Hardware Trojans that leak sensitive information through side channels exploit the performance margins allowed for coping with process variations and rely on systematically induced offsets in the operating point of a fabricated chip, in order to establish a covert channel. Hardware dithering diminishes robustness of such hardware Trojans by challenging their fundamental requirement of a stable operating point, induced offsets from which can encode and leak data. More specifically, through use of calibration knobs which modulate the performances of a fabricated chip, alongside with a random number generator and an NVM which provides a sequence of tuning knob values, hardware dithering sends the circuit on a random walk within the acceptable range of its performances. Thereby, while normal functionality remains unaffected, randomness added to the side channel through which information is being leaked distorts the offsets and, by extension, the leaked data. Effectiveness of hardware dithering was experimentally evaluated through over-the-air measurements using a hardware Trojan leaking the AES key from a wireless cryptographic IC fabricated in a 0.35μm CMOS technology. As demonstrated, while the hardware Trojan operation is perfectly robust in the absence of hardware dithering, activation of the proposed prevention method results in approximately half of the bits in every transmitted packet becoming corrupted. Furthermore, considering that such corruption occurs equiprobably across the bit-positions of a packet, the information theoretic content of the data received through the covert channel is minimized and the Trojan is, effectively, neutralized.

Acknowledgment

This research has been partially supported by the Semiconductor Research Corporation under contract SRC 2625.001 and the National Science Foundation under award NSF 1527460.

References

[1] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using IC fingerprinting. In *IEEE Symposium on Security and Privacy (SP)*, pages 296–310. IEEE, 2007.

[2] A. Antonopoulos, C. Kapatsori, and Y. Makris. Trusted Analog/Mixed-Signal/RF ICs: A Survey and a Perspective. *IEEE Design Test*, 34(6):63–76, Dec 2017.

[3] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor. Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution. *IEEE Design Test*, 30(3):6–17, 2013.

[4] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan. Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247, 2014.

[5] S. Callegari, R. Rovatti, and G. Setti. Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos. *IEEE Transactions on Signal Processing*, 53(2):793–805, 2005.

[6] D. Das, S. Maity, S. B. Nasir, S. Ghosh, A. Raychowdhury, and S. Sen. High efficiency power side-channel attack immunity using noise injection in attenuated signature domain. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67, 2017.

[7] J. He, Y. Zhao, X. Guo, and Y. Jin. Hardware Trojan Detection Through Chip-Free Electromagnetic Side-Channel Statistical Analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–10, 2017.

[8] B. Khaleghi, A. Ahari, H. Asadi, and S. Bayat-Sarmadi. FPGA-Based Protection Scheme against Hardware Trojan Horse Insertion Using Dummy Logic. *IEEE Embedded Systems Letters*, 7(2):46–50, 2015.

[9] F. Koushanfar and A. Mirhoseini. A unified framework for multimodal submodular integrated circuits Trojan detection. *IEEE Transactions on Information Forensics and Security*, 6:162 –174, 2011.

[10] Y. Liu, K. Huang, and Y. Makris. Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting. In *ACM/IEEE Design Automation Conference*, pages 1–6, 2014.

[11] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris. Silicon Demonstration of Hardware Trojan Design and Detection in Wireless Cryptographic ICs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4):1506–1519, 2017.

[12] Y. Liu, G. Volanis, K. Huang, and Y. Makris. Concurrent hardware trojan detection in wireless cryptographic ICs. In *IEEE International Test Conference (ITC)*, pages 1–8, 2015.

[13] Y. Lu, K. S. Subramani, H. Huang, N. Kupp, K. Huang, and Y. Makris. A comparative study of one-shot statistical calibration methods for analog / RF ICs. In *2015 IEEE International Test Conference (ITC)*, pages 1–10, 2015.

[14] C. Maxey, S. Raman, K. Groves, T. Quach, L. Orlando, A. Mattamana, G. Creech, and J. Rockway. Mixed-signal SoCs with in situ self-healing circuitry. *IEEE Design Test of Computers*, 29(6):27–39, 2012.

[15] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia. Hardware Trojan detection by multiple-parameter side-channel analysis. *IEEE Transactions on Computers*, 62(11):2183–2195, 2013.

[16] V. Natarajan, S. Sen, A. Banerjee, A. Chatterjee, G. Srinivasan, and F. Taenzler. Analog signature-driven postmanufacture multidimensional tuning of RF systems. *IEEE Design Test of Computers*, 27(6):6–17, 2010.

[17] B. Sunar, W. J. Martin, and D. R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1):109–119, 2007.

[18] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware Trojans: Lessons Learned After One Decade of Research. *ACM Trans. Des. Autom. Electron. Syst.*, 22(1), May 2016.

[19] K. Xiao and M. Tehranipoor. BISA: Built-in self-authentication for preventing hardware Trojan insertion. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 45–50, 2013.