Concurrent Hardware Trojan Detection in Wireless Cryptographic ICs

Yu Liu*, Georgios Volanis*, Ke Huang†, and Yiorgos Makris*

*Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX 75080
†Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92115

***Abstract*—We introduce a Concurrent Hardware Trojan Detection (CHTD) methodology for wireless cryptographic integrated circuits (ICs), based on continuous extraction of a side-channel fingerprint and evaluation by a trained on-chip neural classifier. While similar statistical side-channel fingerprinting methods have been extensively studied in the past, they operate either before an IC is deployed or, periodically, during idle times, after an IC is deployed. Therefore, they can be easily evaded by a hardware Trojan which remains dormant at all times except during normal operation. In contrast, the proposed methodology operates concurrently with the normal functionality of the IC and is, therefore, much harder to evade. The proposed methodology is demonstrated using a hybrid experimentation platform consisting of (i) a custom-designed wireless cryptographic IC, infested with hardware Trojans that are controllable to be either active or dormant, (ii) a Spice-level simulation model of the fingerprint extraction circuit, and (iii) a custom-designed programmable analog neural network IC. Experimental results corroborate that the proposed CHTD methodology effectively identifies hardware Trojans when they are active, while not incurring any false positives when they are absent or dormant.**

I. Introduction

As semiconductor companies have largely become fabless, with IC manufacturing outsourced to foundries located in low-cost-of-labor parts of the globe, trustworthiness concerns about chips fabricated through this model have ensued. Specifically, ensuring that a manufactured IC has not been subjected to malicious modifications (a.k.a hardware Trojans) capable of undermining its operation, producing erroneous results, or stealing sensitive information, has become a problem of contemporary interest [1], [2]. Accordingly, numerous hardware Trojan detection methods have been recently proposed [3]. Most such methods are applicable during manufacturing testing, prior to IC deployment. A few methods have also followed the Built-in Self-Test (BIST) paradigm, seeking to periodically examine a circuit during idle times after its deployment. All of them, however, can be evaded by hardware Trojans which are active only when an IC performs its normal operation and dormant at all other times.

In an effort to overcome this limitation, herein we propose a concurrent hardware Trojan detection (CHTD) method which is applied continuously and in parallel with normal IC operation. To this end, we borrow the fundamental operating principles of concurrent error detection (CED), a well-studied area for both analog and digital ICs [4], [5], which intends to detect run-time errors instigated by transient, intermittent, or aging-induced sources. CED methods are based on the concept of *invariance*, which is a property that holds true if and only if the circuit operates correctly. Circuitry is added to continuously extract that property during normal operation and check its compliance. The simplest example is *Identity*, which can be extracted through a circuit duplicate and checked through an equality comparator [6]. Other examples include the use of parity and other error detecting codes along with appropriate checkers [7]. While CHTD has a similar objective as CED, namely detection of run-time deviation from trusted operation, its task is harder since it aims at identifying the impact of unknown and carefully hidden culprits rather than modeled and well-understood phenomena. Therefore, assuming that the adversary stages the hardware Trojan attack through an untrusted foundry, we advocate that the invariant property used for CHTD and the criterion employed to evaluate its compliance should not be publicly known. In fact, to prevent the adversary from reverse engineering them from the layout file, their exact details should be withheld from the design and should be introduced and individualized to each chip *after* fabrication, through non-volatile memory (NVM).

II. Concurrent Hardware Trojan Detection

The general idea of the proposed CHTD method, which is based on a popular hardware Trojan detection method, namely statistical side-channel fingerprinting [8]–[14], is shown in Fig. 1. Programmable circuitry is added for extracting and checking an invariant property of the circuit monitored for hardware Trojans alongside its normal operation. The checker, which asserts the CHTD output when the invariance is violated, is implemented as an on-chip one-class classifier. This classifier is individually trained for each chip after fabrication, using

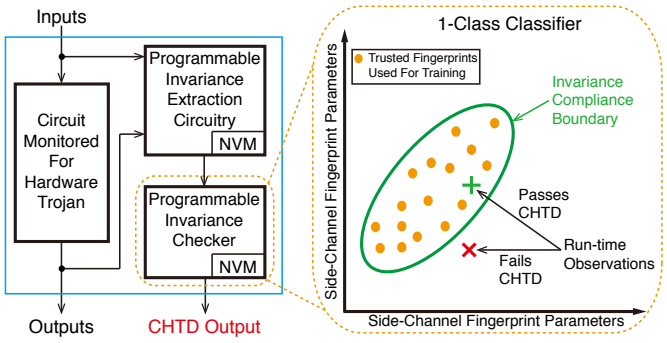

Fig. 1: Concurrent hardware Trojan detection overview

978-1-4673-6578-9/15/$31.00

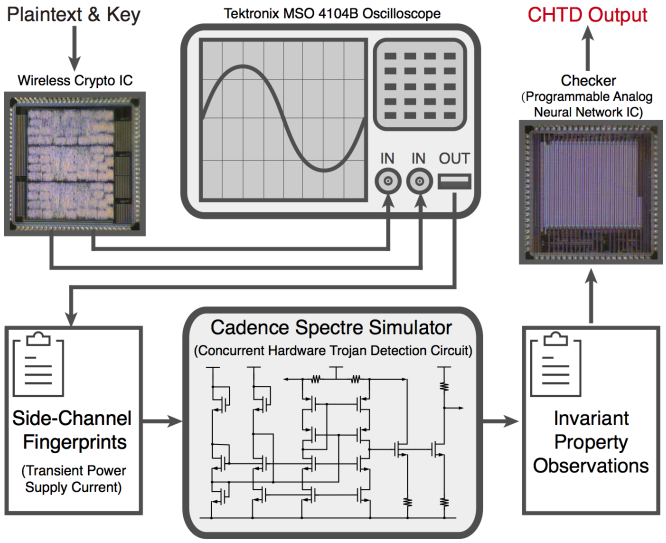

Fig. 2: Experimentation platform

trusted side-channel fingerprints obtained at test time, when the targeted hardware Trojans are dormant. The trained classifier can, then, be used to examine compliance of runtime observations of the invariant property, by comparing their footprint in the side-channel fingerprinting space to the learned boundary.

The underlying premise of this method is that any distortion imposed by an active hardware Trojan on the parametric profile of an IC has to be systematic and above noise-level, in order to be of utility to an adversary. However, by individually training the decision boundary for each chip, the only margins left for a hardware Trojan to exploit are measurement noise and non-idealities of the invariance extraction and checking circuitry and the training algorithm. Therefore, evading detection becomes particularly challenging.

The proposed CHTD method is introduced in the context of wireless cryptographic ICs. Such circuits receive and transmit information, typically encrypted, over public channels. Hence, they constitute an appealing target because they offer a tangible hardware Trojan objective (i.e., to leak secret information) and because no physical access to the actual chips is needed.

III. Experimentation Platform

The experimentation platform which we implemented and used for this study is shown in Fig. 2. Our platform involves Trojan-free and Trojan-infested versions of a wireless cryptographic IC [15], along with external instrumentation for measuring parameters from which an invariant property can be constructed. These measurements are then used as stimuli for performing Spice-level simulation of the circuit that implements the invariant property. The results of the simulation are then passed on to a programmable neural network IC, which implements a classifier (checker) that decides whether the invariant property is violated and drives the CHTD output.

Below, we first introduce the Trojan-free version of the wireless cryptographic IC. Then, we describe the invariant property which we use for the purpose of CHTD, along with its hardware implementation. Next, we introduce the on-chip

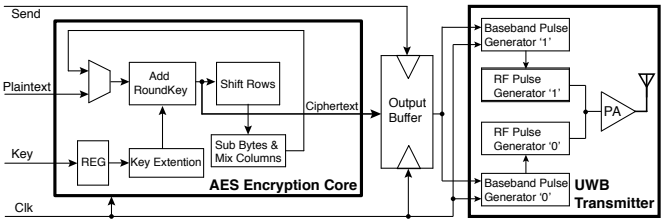

Fig. 3: Block diagram of wireless cryptographic IC

checker which is employed for examining compliance of the invariant property. Lastly, we demonstrate that the CHTD circuitry does not introduce false-positives, i.e. it correctly remains silent in a hardware Trojan-free chip, even in the presence of measurement noise.

A. Wireless Cryptographic IC

Our wireless cryptographic IC includes a digital and an analog part. The digital part consists of an Advanced Encryption Standard (AES) core and an output buffer. The AES core receives plaintext in blocks of 128 bits, which is encrypted using a 128-bit key that is stored on-chip. The width of the encryption key determines the number of transformation rounds to which the plaintext is subjected during encryption. In this case, after 10 rounds of transformation, the plaintext is encrypted into ciphertext, which is stored in the output buffer in blocks of 128 bits, until it is transmitted. The analog part is a small and easy to integrate Ultra-Wide-Band (UWB) transmitter, which includes a baseband pulse generator and an RF pulse generator, as shown in Fig. 3. In our design, frequency-shift keying (FSK) is used to distinguish the polarity of a bit, while on-off keying (OOK) is used to separate adjacent bits. Bit values of '0' and '1' are separated and converted to RZ (return-to-zero) format in the baseband pulse generator. An example of a typical transmission of a '1' and a '0' is shown in Fig. 4. We note that transmission of signal '1' has higher frequency and lower amplitude than transmission of signal '0'. This wireless cryptographic IC was fabricated in TSMC's 0.35μm process, with each die including one Trojan-free version and two Trojan-infested versions, which we discuss further in Section IV.

B. Invariance

In order to identify an invariant property with a high probability of being violated by an active hardware Trojan, we point out that, in our threat model, the only entity an attacker has access to in a wireless cryptographic IC is transmission power. Therefore, hardware Trojans will have to leak information by manipulating the parametric profile (i.e. amplitude, frequency, phase, or combinations thereof) of transmission power. To evade detection, such manipulation needs to be hidden within the margins allowed for process variations, therefore no extra bits can be transmitted and no analog/RF specifications can be violated. In other words, the transmitted signal will appear perfectly legitimate, yet the knowledgeable adversary who knows how it has been manipulated, will be able to extract the leaked information.

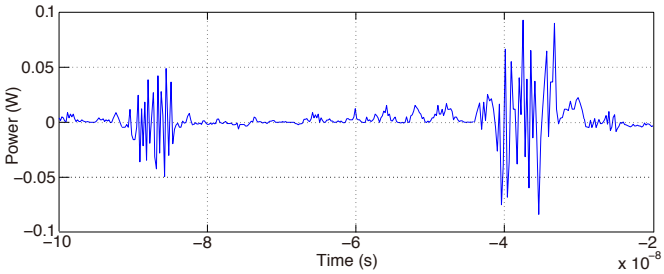

Fig. 4: Transmission power while sending a '1' and a '0'

Manipulating transmission power, however, has a direct impact on the current drawn by the UWB transmitter. Hence, the invariant property that we chose to construct is based on power supply current monitoring[1]. Specifically, we leverage the fact that every time the UWB transmitter transmits a ciphertext bit of '1', it is expected to draw the same current, I_{C_1}, and every time it transmits a ciphertext bit of '0', it is expected to draw the same current, I_{C_0}. Accordingly, we continuously monitor the transient current drawn from the power supply and convert it to voltage, which we then integrate to obtain a value V_{C_1} and V_{C_0} when transmitting a '1' and a '0', respectively. Of course, since these are analog quantities, integrated voltage equality across consecutive transmissions of the same ciphertext bit value is subject to a small difference, which accounts for measurement noise and other non-idealities. Based on the above, our invariant property seeks to ensure such consistency, anticipating that a hardware Trojan which systematically manipulates transmission power in order to leak information will violate the invariance.

We now describe the details of our invariant property. Let $V_{int}(k,m) = m \cdot V_{C_1} + (k-m) \cdot V_{C_0}$ denote the integrated voltage required for transmitting k bits, of which m bits are '1' and $k-m$ bits are '0'. Let us also assume that we monitor the transmitted ciphertext bit-stream and we integrate the voltage for the first m '1s' and the first $k-m$ '0s', resulting in a cumulative integrated voltage $V_A(k,m)$. Note that more than k bits may have been transmitted by the time we meet the above conditions. We then store $V_A(k,m)$ and repeat the process for the next m '1s' and $k-m$ '0s', obtaining a second cumulative integrated voltage reading $V_B(k,m)$. Accordingly, we expect that the following invariance should hold true:

$$|V_A(k,m) - V_B(k,m)| = \delta_{noise} \quad (1)$$

where δ_{noise} reflects measurement noise and non-idealities.

Generalizing further, we can use different k and m values for the two integrated voltage observations. Then, the invariance becomes:

$$|V_A(k_A,m_A) - V_B(k_B,m_B)| = \delta_{noise} + |(m_A - m_B) \cdot V_{C_1} + [(k_A - m_A) - (k_B - m_B)] \cdot V_{C_0}| \quad (2)$$

where the second line of this equation is a constant.

[1]We note that, as is typical in mixed-signal designs, the power supplies of the analog and digital portions of the chip are separate.

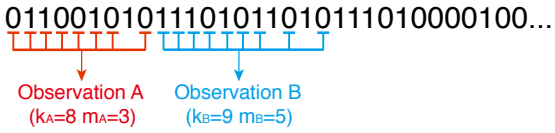

Fig. 5: An example of observation formation

For example, consider the ciphertext bitstream shown in Fig. 5, where $k_A = 8$, $m_A = 3$, $k_B = 9$ and $m_B = 5$. The figure shows the bits during transmission of which the voltage is integrated for each of the two observations, A and B, respectively. Based on Equation 2, in a Trojan-free circuit the absolute difference between observations $V_A(8,3)$ and $V_B(9,5)$ should invariably be $1 \cdot V_{C_0} - 2 \cdot V_{C_1} + \delta_{noise}$. A checker capable of evaluating this invariance can, then, be used to detect run-time violations by active hardware Trojans.

Before we describe the circuitry implementing the above invariant property and the checker, we make three important observations:

1) Checking is performed non-intrusively, along with the circuit operation, and is continuously repeated every time the voltage integration conditions are satisfied two consecutive times.
2) This is a *self-referencing* approach [10], which makes it very difficult for a hardware Trojan to evade it, since the available margin wherein it must hide its impact is below the noise level; therefore, even the knowledgeable adversary will be unable to robustly distinguish the leaked information from noise.
3) The exact values of k_A, k_B, m_A, m_B, and δ_{noise} may be different for each chip and may be decided and programmed *after* fabrication through non-volatile memory; this uncertainty makes it very difficult for an attacker to design a hardware Trojan which can evade the invariant property checking.

C. Hardware Implementation

In Fig. 6, we show the circuitry that measures the integrated voltages for the two observations, $V_A(k_A,m_A)$ and $V_B(k_B,m_B)$, which we need for evaluating the invariance. The right part of the figure shows the analog components which are needed for supply current sensing, conversion to voltage, integration and storage of each observation. Details regarding the Current Sensor and the Voltage Integrator components are provided at the end of this subsection. The left part of the figure shows the digital control logic, wherein a 2-bit counter directs the circuit through its three main states:

1) **Collect Observation A:** In its starting state of '00', the 2-bit counter instructs the two MUXes to select m_A and $k_A - m_A$ as the values passed to the two equality comparators. These comparators indicate when the count of '1s' has reached the target m_A and when the count of '0s' has reached the target $k_A - m_A$. These counts are kept by the two counters shown below the equality comparators, which count upwards every time

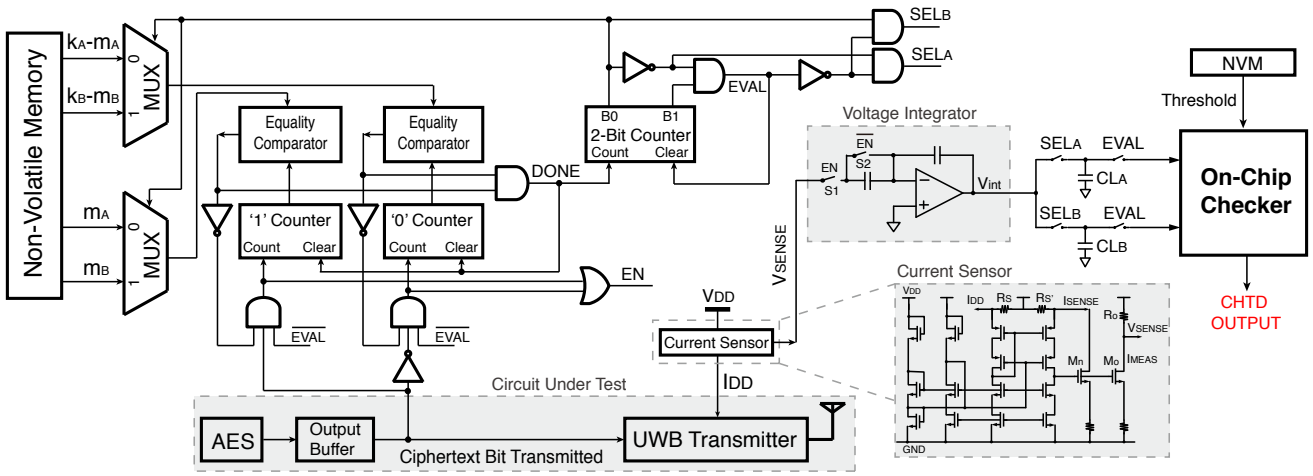

Fig. 6: Hardware implementation of invariant property for CHTD

the ciphertext bit currently transmitted by the UWB transmitter is a '1' or a '0', respectively. Counting stops when the corresponding target is reached and when we are in the evaluation state. When either a '1' or a '0' in the ciphertext bit-stream is counted towards the targets m_A and $k_A - m_A$, respectively, the signal EN is asserted to indicate that the current sensor and voltage integrator should be active during the transmission of this ciphertext bit, thereby contributing to observation $V_A(k_A, m_A)$ which is stored in the load capacitor CL_A by enabling SEL_A. When both equality comparators reach their target, the signal $DONE$ is asserted. This prompts the two counters to reset and pushes the 2-bit counter to the next state, '01'.

2) **Collect Observation B:** In the '01' state, the 2-bit counter instructs the two MUXes to select m_B and $k_B - m_B$ as the values passed to the two equality comparators. Everything else operates exactly as in the first state with two main differences: (i) when the signal EN is asserted, the integrated voltage contributes to observation $V_B(k_B, m_B)$ which is stored in the load capacitor CL_B by enabling SEL_B, and (ii) counting stops when m_B '1s' and $k_B - m_B$ '0s' have been encountered. When both equality comparators reach their target, the signal $DONE$ is asserted again, prompting the two counters to reset and pushing the 2-bit counter to the next state, '10'.
3) **Evaluate Invariance:** In the '10' state, the $EVAL$ signal is asserted so both SEL_A and SEL_B are off. In this way, the two observations $V_A(k_A, m_A)$ and $V_B(k_B, m_B)$ are passed on to the on-chip checker by fully discharging the two load capacitors CL_A and CL_B, wherein they are stored, respectively. This process finishes in less than a clock cycle, so the 2-bit FSM is immediately reset to '00' in the next clock cycle, through its synchronous $CLEAR$ input.

Current Sensor: The schematic of the CMOS built-in current sensor (BICS) is shown in Fig. 6. This design is aimed to radio frequency applications [16] and it is used to monitor the power supply current drawn by the UWB transmitter and convert it to a voltage signal, V_{sense}, which is defined as:

$$V_{sense} = V_{DD} - \frac{(W/L)_{M_o}}{(W/L)_{M_n}} \cdot \frac{R'_s}{R_s} \cdot I_{DD} \cdot R_o \qquad (3)$$

The operating point of V_{sense} is controlled by the ratio of the sizes of transistors M_o and M_n, the ratio of resistors R'_s and R_s, and resistance R_o. R'_s and R_s should be very small so that the supply voltage drop remains very small and does not affect the normal operation of the UWB transmitter. In our design, a 10-Ohm metal wire resistor proved sufficient for sensing the transient current.

Voltage Integrator: The output of the current sensor, V_{sense}, is connected to the input of the voltage integrator, the schematic of which is also shown in Fig. 6. When EN is high, $S1$ turns on and $S2$ turns off, so the circuit enters the voltage integration mode. When EN is low, $S1$ turns off and $S2$ turns on, so the integrator enters the hold mode, in which the integrated voltage is retained.

Load Capacitors: Fig. 6 also shows the two load capacitors, CL_A and CL_B, which are used to store the integrated voltages for the two observations, $V_A(k_A, m_A)$ and $V_B(k_B, m_B)$, respectively. In the first state, SEL_A is on and SEL_B is off, therefore the voltage integrator charges CL_A. In the second state, SEL_A is off and SEL_B is on, therefore the voltage integrator charges CL_B. In both cases, the $EVAL$ signal is '0'. In the third state, both SEL_A and SEL_B are turned off, but $EVAL$ is '1' so the two observations are passed on to the checker for evaluating the invariance, as we describe next.

D. Invariance Checker

Checking the invariance requires an on-chip circuit that evaluates the two observations $V_A(k_A, m_A)$ and $V_B(k_B, m_B)$ for compliance to Equation 2. Evidently, this on-chip circuit will have to be programmable after fabrication, in order to account not only for the chosen values of k_A, k_B, m_A, and m_B but also for the slight differences in V_{C_0} and V_{C_1}

among different chips, which are caused by manufacturing process variations. To this end, we propose the use of an on-chip 1-class neural classifier as our invariance checker. Using trusted observation pairs obtained prior to deployment, when a hardware Trojan is either absent or dormant, we train this 1-class classifier to learn the boundary that encloses Trojan-free pairs in the two dimensional space of observations. This boundary is, then, used to evaluate the observation pairs that are obtained during normal operation after the chip is deployed, asserting the CHTD output when an observation pair that falls outside this boundary is encountered.

In this work, we employ a custom-designed programmable analog neural network experimentation platform [17], which is shown in Fig. 7(a). This chip consists of a reconfigurable 30x20 array of synapses and neurons along with the needed peripheral circuits. The synapse circuits employ two modes of weight storage: (i) dynamic for fast updating of their weights using capacitors during training and, (ii) nonvolatile, for permanent storage of the learned weights using floating gate transistors (FGTs) after the training is completed. The synapse implements a multiplication between the differential currents of the weights and input signals. The results of these multiplications are summed and fed to the corresponding neuron, which implements a sigmoid activation function. The neuron in the last layer produces the output of the neural network. There are also three peripheral circuits. The differential transconductors (GM) accept as input voltage-encoded signals (i.e., the output of the voltage integrator) and perform a voltage-to-differential current conversion which is needed for the core of the circuit. The digitally-controlled current source (DCCS) generates target currents (Iprog) for dynamic memory programming. Finally, the current to voltage converter ITOV is used for the reading of the internal currents.

The chip is fabricated in TSMC's 0.35μm process and supports implementation of various topologies which can learn very complex boundaries. Among them, in this work we experimented with the multilayer perceptron (MLP) learning model, a 2-input instance of which is shown in Fig 7(b). The MLP is a simple and easy to train feed-forward network; it does not contain feedback loops and each layer receives connections only from inputs or previous layers. The first layer (also known as hidden) has 3 neurons which receives the two observations at its inputs X_1 and X_2 along with a constant bias $X_0 = 1$. The second layer (also known as output) contains a single neuron (for binary classification) which receives the outputs Y_1^H, Y_2^H and Y_3^H of the first layer and produces the network output Y^O. The strength of connections is controlled by synapses, which act as multipliers of input signals and their local weight values. The sum of synaptic products is passed through a nonlinear activation function of a neuron. The number of input and output neurons depends on the classification problem, while the number of hidden neurons reflects the learning capacity of the model. In our case, a small number of hidden neurons (i.e., 3) is sufficient for learning the boundary that encloses the observations that comply with the invariant property.

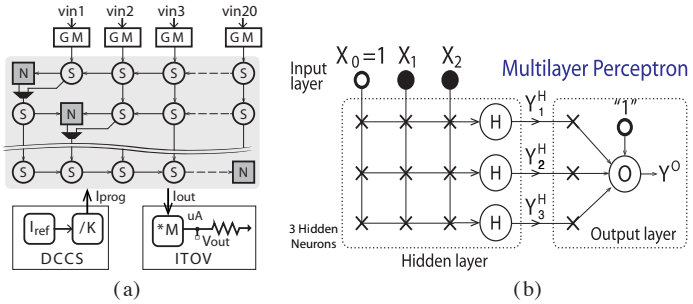

Fig. 7: (a) programmable analog neural network IC, (b) MLP

At this point, we should note that only a *small fraction* of the resources available on this programmable analog neural network IC need to be integrated on each die for CHTD. In our experiment, learning the appropriate boundary for checking the invariant property required 3 hidden neurons, the output neuron, and a total of 13 synapses, as shown in Fig 7(b). Along with the three transconductors (GM) needed for the two inputs and the bias, these are the only components that need to be integrated on-chip for CHTD. The peripheral circuits required for programming the neural network do not have to be on the same die, as no in-field reprogramming is needed. Overall, the analog nature of the neural network makes for a very flexible yet compact and low power implementation of the checker. Finally, we point out that the synapse weights learned during training are stored as an analog charge quantity in a non-volatile fashion in the FGTs included in the synapses, with the current platform supporting up to 10-bits of precision for each weight.

E. Measurements

We now demonstrate that the proposed method for extracting and checking the invariant property does not introduce false positives in Trojan-free circuits. Using the hybrid experimentation platform of Fig. 2 and one of the fabricated wireless cryptographic ICs we measured and recorded the transient power supply current consumption waveforms while transmitting a randomly generated ciphertext bit-stream. We then used these waveforms to perform SPICE-level simulation of the invariant property extraction circuitry shown in Fig. 6, with $k_A = 8$, $m_A = 2$, $k_B = 8$, and $m_B = 4$, and we used the first 50 pairs of invariance observations $V_A(8, 2)$ and $V_B(8, 4)$ to train the neural classifier described in the previous subsection. We then repeated this experiment in different locations and under minor perturbations in operating conditions (i.e., Vdd and temperature), in order to account for the impact of measurement noise. Thereby, we collected another 50 pairs of invariance observations $V_A(8, 2)$ and $V_B(8, 4)$, which we used as the validation set for our trained classifier. The results are shown in Fig. 8, where the actual boundary learned by the trained 1-class classifier successfully encloses both the training and the validation set. This result remained valid in all cross-validation iterations and for every combination of k_A, k_B, m_A, and m_B that we experimented with. This corroborates that,

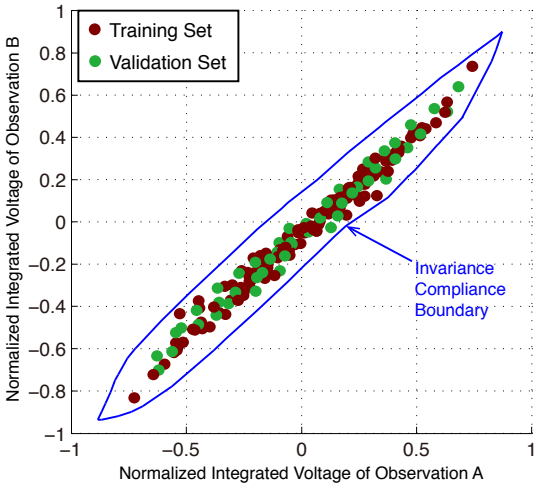

Fig. 8: Boundary learned by trained 1-class neural network

even in the presence of measurement noise, the invariance extraction circuit and the trained classifier operate correctly and do not inadvertently assert the CHTD output for a Trojan-free circuit (i.e., no false positives).

IV. CHTD Effectiveness Evaluation

In order to evaluate effectiveness of our CHTD method in detecting Trojans, we use two Trojan-infested versions of the wireless cryptographic IC shown in Fig. 3, integrated on the same die as the Trojan-free version [15]. The two Trojan-infested circuits leak the AES encryption key, one bit at a time, by using its value to slightly modulate the amplitude or the frequency of wireless transmission power, as shown in Figs. 9 (a) and (b), respectively. Specifically, when the leaked key bit is '1', the amplitude/frequency are left unaltered, while when the leaked key bit is '0', they are slightly increased. This increase, which is 8mW or 14mW for the first Trojan and 0.128GHz or 0.039GHz for the second Trojan, when transmitting a ciphertext bit of '1' or '0', respectively, is within the specification margins allowed for process variations; hence the contaminated waveforms continue to pass all circuit specifications. While the unsuspecting user may attribute the incurred transmission power variance to noise, the knowledgeable adversary who knows how the leaked information is systematically encoded on this variance will be able to retrieve it. An external control allows us to set each of the two hardware Trojans to dormant or active state, thereby facilitating our CHTD study.

For the purpose of our experiment, we used a randomly generated 128-bit AES key and a randomly generated plaintext, which was encrypted into a ciphertext by the AES module. For each of the two hardware Trojan-infested circuits, we first put the Trojan in the dormant state and recorded the first 100 observation pairs $V_A(8,2)$ and $V_B(8,4)$ produced by the invariance extraction circuit, which required transmission of 2293 bits of ciphertext. We then used these observations to train the one-class neural network. Finally, we repeated the

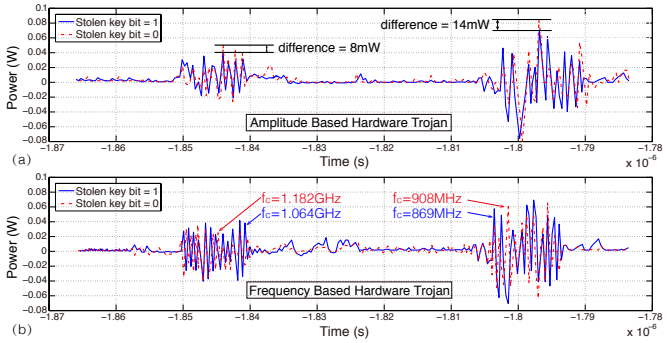

Fig. 9: Transmission power waveforms of Trojan-infested ICs

same experiment, this time with the hardware Trojan activated. Once again, we recorded the first 100 observation pairs $V_A(8,2)$ and $V_B(8,4)$ produced by the invariance extraction circuit and we evaluated them by the trained classifier. The results for the two Trojan-infested circuits are shown in Figs. 10 and 11, respectively.

As can be seen, in both cases, all 100 invariance observations are enclosed within the learned boundary when the hardware Trojan is dormant. When it is active, 94 of the 100 invariance observations fall outside the boundary and are immediately caught by the trained classifier. However, there exist 6 instances where, in both cases, the footprint of the invariance observation is inside the boundary, even though the hardware Trojan is active. Upon further analysis, we observed that, in all 6 of these cases, the following property was true: the key bits leaked when taking observation V_A and the key bits leaked when taking observation V_B had the *same number of '0s'*. Indeed, these two hardware Trojans slightly increase the integrated voltage of our invariance observations by a small but constant amount when the leaked key bit is a '0' and have no impact on it when the leaked key bit is '1'; hence, the above property *masks* their impact because the left hand side of Equation 2 remains the same.

Fortunately, this masking is only temporary. While the probability that the key bits leaked during observation V_A and observation V_B have the same number of '0s' is non-negligible, the probability that this will be the case *every time the invariance is checked* is infinitesimal. In fact, in our experiment, the next invariance check following each of the six misclassified instances failed the invariance compliance test. In other words, the hardware Trojan is still detected for these cases, but with a latency of one invariance check, or approximately 23 bits of ciphertext transmission in our case.

V. Discussion

The following points elaborate on a few key aspects off the proposed CHTD method, in order to facilitate a comprehensive understanding of its capabilities and limitations.

- **Overhead:** Our current experimentation platform shown in Fig. 2 is a hybrid involving two chips and Spice-level simulation using a post-layout extracted model of the invariance extraction circuitry. Therefore, in order

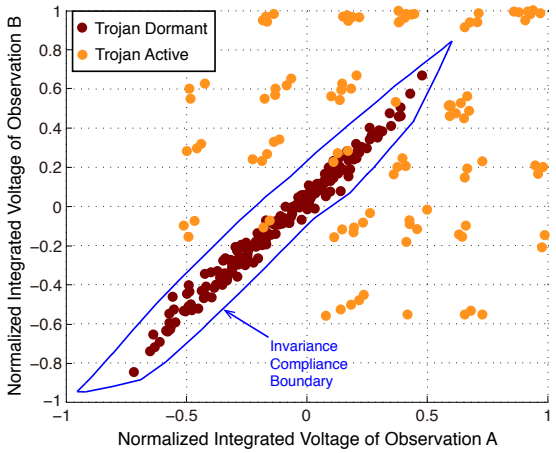

Fig. 10: CHTD results for amplitude-based hardware Trojan

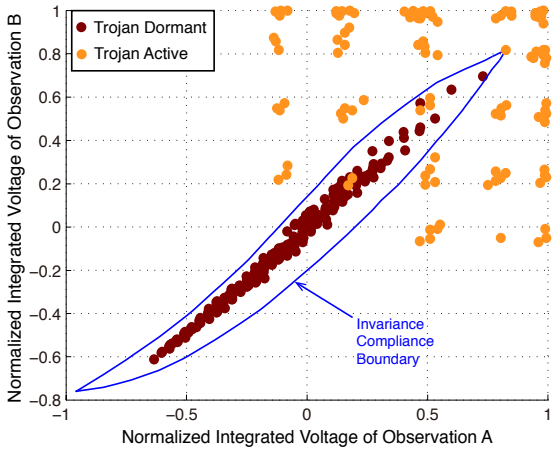

Fig. 11: CHTD results for frequency-based hardware Trojan

to obtain an accurate overhead approximation, we first measured the power of the wireless cryptographic IC and the neural network chip, while transmitting 100 blocks of 128-bits and checking the invariance. We then added the power consumption of the invariance extraction circuitry, as reported by Spice simulation, for the same transmission. The results place the power overhead of the CHTD circuitry at approximately 6%. Area overhead was computed similarly and is approximately 9% of the baseline design. Given the importance of ensuring run-time trustworthiness of ICs deployed in sensitive applications, we feel that such single-digit percentile overhead is a worthy investment.

- **Threat Model:** The proposed solution seeks to detect hardware Trojans which are activated only in the field of operation. As such, it should not be viewed as a replacement of design-time or pre-deployment hardware Trojan detection methods but rather as complementary to them. We also note that the proposed solution is independent of where, when and how the hardware Trojan is implanted. Indeed, since this is a self-referencing approach, it does not require availability of a "golden" model but, rather, relies on run-time inconsistencies to detect hardware Trojan activation. Finally, we emphasize that the proposed method is particularly geared towards covert attacks which are hidden in the parametric space and which *do not alter functionality*. Hardware Trojans with overt payload, such as using the legitimate public channel to openly transmit plaintext or ciphertext encrypted with an attacker-modified key are easily detectable through functional test approaches in a supervised setting.
- **Trojan-Agnostic Solution:** We note that our one-class classifier is trained using only measurements from Trojan-dormant circuits and that the proposed CHTD method does not require any knowledge of the hardware Trojan operating principles. The only underlying assumption is that a hardware Trojan in wireless cryptographic ICs will have to distort transmission power, which is the only parameter an attacker has access to, and by extension, the power supply current profile of the transmitter. Accordingly, the chosen invariant property, which reflects this profile, should be able to detect any hardware Trojan that distorts transmission power, independent of how it encodes the leaked information.
- **Addressing Single-Point-of-Failure:** Similar to a stuck-at-0 fault at the CED output, which will prevent a CED method from reporting any errors, a hardware Trojan attack which disables CHTD when the Trojan is activated would counteract its operation. A possible solution to this single-point-of-failure limitation would require to intentionally make the CHTD circuit fail in a supervised setting, through means that the attacker has no way of detecting. For example, one could intentionally vary V_{dd} beyond the expected acceptable range, forcing the invariant property to be violated and, hence, expecting CHTD to indicate an error. A suppressed CHTD would fail to do so, thereby exposing the attack. A similar approach can be used during testing of the CHTD circuit, in order to expose manufacturing defects.

VI. Comparison to Related Work

Only a handful of methods exist in the literature for detecting hardware Trojans in parallel with the normal operation of an IC. In [18], the authors propose compilation of functionally equivalent variants of a process and dynamic comparison of their execution across different cores. Besides the incurred performance overhead, this technique is limited to multi-core systems and requires support from the software/OS. The method proposed in [19] utilizes thermal sensors to detect run-time deviations from the trusted power/thermal profile, which could be caused by Trojan activation. However, a small, sophisticated Trojan design which consumes a small amount of current and is only active for small amounts of time, may evade detection since the thermal profile is a slowly changing parameter. Closer to the work proposed herein, a current sensor and a comparator are used in [20] and [21] to monitor the transient power supply current and compare it

to a reference value, in order to distinguish between Trojan-free and Trojan-infested circuits. A potential problem with the first of these two methods is that the comparison threshold is the same and is statically defined for all chips, allowing enough margin to account for process variations. Therefore, the attacker might be able to reverse-engineer the reference value used by the comparator and design the hardware Trojan so that its impact remains within the comparison threshold. The second of these methods individualizes the reference per chip and sets it after manufacturing, through NVM, thereby avoiding the above problem. However, this method is only applied on a Linear Feedback Shift Register (LFSR) which performs a very specific function (i.e., pseudo-random number generation) and has no inputs during its normal functionality. Generalizing this method to arbitrary circuits requires adaptation of the threshold to the inputs of a circuit, which is very challenging, if at all possible, to do with a simple comparator. Finally, we should note that all of these methods target digital circuits. To our knowledge, the method proposed herein is the first concurrent hardware Trojan detection approach in mixed-signal/analog/RF ICs. The closest method that we are aware of in this domain is [22], which is not concurrent but rather applied periodically during idle times, after IC deployment.

VII. Conclusions

Detection of hardware Trojans which are active only when an IC is performing its normal operation requires a runtime monitoring approach. To this end, the method proposed herein extracts invariant side-channel fingerprints of a circuit concurrently with its normal operation and statistically evaluates their compliance through a trained on-chip classifier. As demonstrated experimentally using a wireless cryptographic IC and an analog neural network experimentation IC, the proposed CHTD method correctly remains silent when hardware Trojans are absent or dormant, yet effectively alerts of the presence of hardware Trojans when they are active. Accordingly, our next-step plans involve monolithic integration of all components and demonstration of this CHTD method on a single-chip, as well as its application and evaluation using other ICs.

VIII. Acknowledgments

This research was partially supported by the National Science Foundation (NSF 1149465: *THWART: Trojan Hardware in Wireless ICs: Analysis & Remedies for Trust*).

References

[1] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[2] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[3] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[4] H.-G. D. Stratigopoulos and Y. Makris, "Concurrent detection of erroneous responses in linear analog circuits," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 25, no. 5, pp. 878–891, 2006.

[5] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?," in *IEEE International Test Conference*, 2000, pp. 985–994.

[6] A. Avizienis and J. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *IEEE Computer*, vol. 17, no. 8, pp. 67–80, 1984.

[7] M. Gossel and S. Graf, *Error Detection Circuits*, McGraw-Hill, 1993.

[8] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symposium on Security and Privacy*, 2007, pp. 296–310.

[9] Y. Jin and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 51–57.

[10] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A scalable side-channel approach for hardware Trojan detection," in *Cryptographic Hardware and Embedded Systems*, pp. 173–187. 2010.

[11] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar, "High-sensitivity hardware Trojan detection using multimodal characterization power mapping of integrated circuits using AC-based thermography," *IEEE/ACM Design, Automation and Test in Europe*, 2013.

[12] Y. Jin and Y. Makris, "Hardware Trojans in wireless cryptographic ICs," *IEEE Design and Test of Computers*, vol. 27, pp. 26–35, 2010.

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

[14] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2010, pp. 13–18.

[15] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ICs: Silicon demonstration & detection method evaluation," in *IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 399–404.

[16] M. Cimino, H. Lapuyade, M. De Matos, T. Taris, Y. Deval, and J.B. Begueret, "A robust 130nm-CMOS built-in current sensor dedicated to RF applications," in *IEEE European Test Symposium*, 2006, pp. 151–158.

[17] D. Maliuk and Y. Makris, "An experimentation platform for on-chip integration of analog neural networks: A pathway to trusted and robust analog/RF ICs," *IEEE Transactions on Neural Networks and Learning Systems (Open Access Pre-Print Available On-line)*, 2015.

[18] D. McIntyre, F. Wolff, C. Papachristou, and S. Bhunia, "Dynamic evaluation of hardware trust," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2009, pp. 108–111.

[19] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 532–539.

[20] Y. Cao, C. Chang, and S. Chen, "Cluster-based distributed active current timer for hardware Trojan detection," in *IEEE International Symposium on Circuits and Systems*, 2013, pp. 1010–1013.

[21] Y. Jin and D. Sullivan, "Real-time trust evaluation in integrated circuits," in *IEEE/ACM Design, Automation and Test in Europe Conference*, 2014, pp. 1–6.

[22] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ICs," in *IEEE/ACM Design, Automation Test in Europe Conference*, 2012, pp. 965–970.