DISTROY: Detecting Integrated Circuit Trojans with Compressive Measurements

Youngjune L. Gwon, H. T. Kung and Dario Vlah
Harvard University
{gyj,htk,dario}@eecs.harvard.edu

Abstract

Detecting Trojans in an integrated circuit (IC) is an important but hard problem. A Trojan is malicious hardware—it can be extremely small in size and dormant until triggered by some unknown circuit state. To allow wake-up, a Trojan could draw a minimal amount of power, for example, to run a clock or a state machine, or to monitor a triggering event. We introduce DISTROY (Discover Trojan), a new approach that can efficiently and reliably detect extremely small background power leakage that a Trojan creates and as a result, we can detect the Trojan. We formulate our method based on compressive sensing, a recent advance in signal processing, which can recover a signal using the number of measurements approximately proportional to its sparsity rather than size. We argue that circuit states in which the Trojan background power consumption stands out are rare, and thus sparse, so that we can apply compressive sensing. We describe how this is done in DISTROY so as to afford sufficient measurement statistics to detect the presence of Trojans. Finally, we present our initial simulation results that validate DISTROY and discuss the impact of our work in the field of hardware security.

1 Introduction

Many semiconductor companies today are fabless. They outsource the manufacturing of their integrated circuit (IC) products to cheaper or more advanced fabrication facilities. While this *go-with-remote-foundries* model provides a compelling option, it makes easier for an attacker to compromise the fabrication process and insert a *Trojan*, malicious hardware that not only alters the original design but also performs security attacks.

Trojans comprise a subtle addition to an IC. A Trojan can be as small as a single gate, or as large as a microcontroller capable of launching systematic security attacks [12]. Detecting Trojans is challenging partly because their structure is unknown, which makes it infeasible to perform functional tests of the IC to detect the Trojans based on the circuit functionality. Trojans are also dormant at times, and in general, there is no *a priori* knowledge of their activation mechanism.

The two premises available to a Trojan detector is that a Trojan will draw some power [1] and alter the physical structure of the circuit [2]. However, detecting the latter–circuit alteration–is difficult and often impractical as it may require costly destructive inspection of the circuit, performed using expensive equipment. Furthermore, a Trojan designer can anticipate and make sure that circuit analysis techniques such as path-length measurement will fail to reveal the Trojan. Therefore, we are left with power usage as the main detection vector.

We assume that a Trojan designer's only option is to increase the power consumption of the circuit. We argue that this is a reasonable assumption based on the following two cases:

1. The Trojan designer does not know the circuit design. In this case, the designer will merely add Trojan gates without modifying the original circuit.
2. The Trojan designer knows the circuit design. In this case, the designer can first optimize the circuit to reduce its power consumption, and then insert the Trojan gates, leaving the power consumption unchanged. However, we can assume this case is avoidable by making the original circuit slack-free, so that no further power reduction is possible.

In this paper we focus on combinatorial circuits, that is, circuits where the inputs and outputs of each gate are determined by the inputs to the circuit. Such circuits may serve as basic blocks of larger, possibly stateful circuits, and can be gated–selectively powered on–to allow testing in isolation. In contrast, we don't restrict the Trojan circuits to be combinatorial; a Trojan may, for instance, run a clock, a state machine, or monitor triggering events for its activation.

Logic gates consume orders of magnitude different amounts of power depending on their inputs. Thus, to expose the Trojan power consumption, it is important to discover circuit inputs, or *test vectors*, which lead to low circuit power consumption by, for example, putting as many gates as possible into low power states. This is generally a hard problem with a number of heuristic solutions based on solving instances of SAT (satisfiability) [10]. We will call such low power consumption-inducing inputs the *revealing* test vectors. In signal processing parlance, we say that with such test vectors the SNR is high, where the desired signal is the Trojan power consumption, and the noise the deviation in power use from expected in the non-Trojan parts of the circuit.

We propose DISTROY (Discover Trojan), a new approach that substantially reduces the I/O requirement in detecting small power leakage due to Trojans embedded in ICs. With reduced I/O, the approach still allows off-chip Trojan detectors to recover the most significant indicators of Trojan-induced power variations. DISTROY relies on the assumption that revealing test vectors are rare, or sparse. Compressive sensing [3], a recent technique developed in signal processing, forms the basis of DISTROY that enables simple encoding and accurate reconstruction of the most significant power consumption anomalies among the test vectors applied. We show that DISTROY can robustly detect the presence of power leakage resulting from the background power consumption of on-chip Trojans even when the drawn power is extremely small.

2 Compressive Sensing

Baraniuk [3] and Candès and Wakin [4] provide good tutorial introductions of compressive sensing for interested readers. We provide here some basic results required to follow the technical details of the paper.

Consider a real-valued, length-N input vector $\mathbf{x} = \langle x_1 x_2 \ldots x_N \rangle$. Suppose the vector has an alternative representation in a basis $\mathbf{\Psi}$, $\mathbf{x} = \mathbf{\Psi s}$, where only K elements of $\mathbf{s}$ are non-zero. We will say that such a vector $\mathbf{x}$ is K-sparse. When $K \ll N$, we regard $\mathbf{x}$ as *compressible*.

Compressive sensing encodes $\mathbf{x}$ by producing *measurements* $\mathbf{y} = \mathbf{\Phi x}$. Here, the matrix $\mathbf{\Phi}$ of size $M \times N$ is called a *measurement matrix*. We can reconstruct $\mathbf{x}$ by solving the system of equations $\mathbf{y} = \mathbf{\Phi x}$ where there are more unknowns (N) than equations (M). Note that in compressive sensing, the number of equations is the number of (compressive) measurements.

Compressive sensing theory states that we can restore $\mathbf{s}$, the K-sparse form of $\mathbf{x}$, with high probability when $\mathbf{\Phi}$ is a random matrix and when $M \geq cK \log(N/K)$ where c is a small constant. Linear programming solves for $\mathbf{x}$ by minimizing the ℓ_1-norm of $\mathbf{s}$:

$$\min_{\mathbf{s} \in \mathbb{R}^N} \|\mathbf{s}\|_{\ell_1} \quad \text{subject to} \quad \mathbf{y} = \mathbf{\Phi x}, \mathbf{x} = \mathbf{\Psi s}. \qquad (1)$$

An interesting property of the ℓ_1-minimization is that the quality of the decoding is a function of M. The larger M is, the more accurate the reconstruction becomes. Furthermore, recovery is incremental—using small M we recover the largest components of $\mathbf{s}$, and if we wish to recover more components, we grow M accordingly.

Another powerful feature of compressive sensing is the *care-free*, low-complexity encoding unlike conventional coding or compression schemes. It is coupled with the flexibility of incorporating any $\mathbf{\Psi}$ in decoding that transforms $\mathbf{x}$ to a sparse form, which makes compressive sensing potentially a ground-breaking solution for many security problems such as intrusion detection and identification of spam and DDoS attacks (or more generically any form of anomaly).

Table 1: Leakage current mean and standard deviation of a 2-input NAND gate (source: Singh *et al.* [10]).

Input (state)	μ (nA)	σ (nA)
00	.223	.082
01 or 10	4.578	3.026
11	13.109	16.785

3 How Trojans May Be Detected

In this section we provide some background on circuit characterization based on power consumption statistics and describe a simple, baseline method for Trojan detection.

3.1 Background: Log-normal Leakage Current Model

ICs typically operate at a fixed voltage. Since power is a product of voltage and current, we will refer to current and power interchangeably. When a circuit's inputs are held constant, the circuit still consumes a certain amount of power because logic gates typically pass a small, but non-zero amount of current. The current consumed in such a static circuit is referred to as *leakage current.*

The leakage current of a gate depends on its inputs. For example, Table 1 shows the leakage current values for a 2-input NAND gate manufactured in a certain process, for 3 different input combinations. We will refer to each input combination as a *gate state*. Note that the leakage currents of different gate states can vary by orders of magnitude! The leakage current varies from gate to gate, because the physical dimensions of gate features vary in manufacturing. Since the physical variations are typically normally distributed, and have an exponential effect on current, a common way to model leakage current is using the *log-normal distribution* [5].

To predict the total power consumption of a circuit, we must know the inputs to each gate, which, for a combinatorial circuit, can be derived from the circuit inputs. For example, consider a circuit depicted in Figure 1. There will be $3 \cdot 3 = 9$ possible combinations of gate states and $2^3 = 8$ test vectors (over input bits X, Y, and Z). Note that circuits often have some gate states that are physically unrealizable. We can then obtain the leakage current distribution for each test vector as the distribution of a sum of log-normal random variables, each corresponding to one gate in a specific state. For example, consider applying test vector $\langle 0, 0, 0 \rangle$ to

Figure 1's circuit, that is, $X = 0, Y = 0, Z = 0$. It follows that gate A will have inputs $\langle 0, 0\rangle$ and gate B inputs $\langle 1, 0\rangle$, and so we can expect that these gates will draw log-normally distributed amounts of current with parameters from the first two rows in Table 1, respectively. We can see that the total current consumed by the circuit in this example will be the random variable $I_{TOTAL} = LN(0.223, 0.082) + LN(4.578, 3.026)$ where $LN(\mu, \sigma)$ denotes a log-normally distributed random variable with the given parameters.

At present, no closed-form is known for the probability distribution of I_{TOTAL}. However, there are many approximation methods which work well in practice, such as that by Fernandes and Vemuri [5], which we adopted for this paper.

3.2 Baseline Approach

Figure 2 depicts a baseline approach for detecting Trojans. We apply N test vectors $v_1, v_2, \ldots, v_N$ to the CUT, obtaining N power measurements $x_1, x_2, \ldots, x_N$, one for each vector. For the same N test vectors, we compute, as outlined in the previous subsection, the expected values of the leakage current distribution for each test vector, $\mathbf{x_G} = \langle g_1, g_2, \ldots, g_N\rangle$. Note that these ground-truth values can be obtained offline.

Next, we compare the power measurements x_i with the expected power measurements to decide whether or not Trojans are present in the CUT. When there are no Trojans present, any deviation from the expected measurement consists of only one source—the chip fabrication variations, which is accounted for by the gate models. However, when there are Trojans present, the gates comprising the Trojan draw additional leakage current and thus shift the probability distribution of total circuit current as depicted schematically in Figure 3. We choose a rejection threshold α such that if the total current is above α then we declare that the circuit contains Trojans.

There always exists some likelihood of error. Referring to Figure 3, we can see errors could occur either 1) when there is no Trojan, but total leakage current is larger than the rejection threshold–an event called a *false positive*, or 2) when there is a Trojan but the total leakage current falls below the threshold, an event called a *false negative.* We can reduce the likelihood of both of these events by testing groups of multiple chips as described in the following subsection.

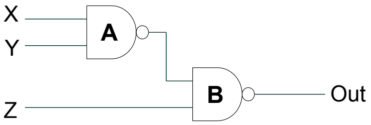
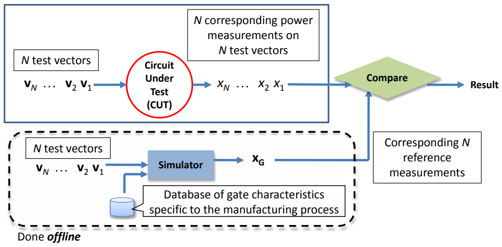


Figure 1: Example circuit with 2 NAND gates.



N test vectors
$\mathbf{v}_N$ … $\mathbf{v}_2$ $\mathbf{v}_1$
Circuit Under Test (CUT)
N corresponding power measurements on N test vectors
x_N … x_2 x_1
Compare
Result
N test vectors
$\mathbf{v}_N$ … $\mathbf{v}_2$ $\mathbf{v}_1$
Simulator
$\mathbf{x}_G$
Corresponding N reference measurements
Database of gate characteristics specific to the manufacturing process
Done offline

Figure 2: Baseline approach.

3.2.1 Testing Multiple Chips from the Same Process to Improve Detection Reliability

We can improve the detection reliability by testing *multiple chips* manufactured with the same fabrication process. We consider the following two ways to use multiple chips, which reduce the false positive and negative rates, respectively:

1) Reducing False Positives. To reduce false positives, we can test groups of P chips, for some P greater than one, and require that for all of them, the total leakage current exceeds the rejection threshold before we can declare a Trojan. The larger P is, the fewer false positives are expected. Note that increasing P also increases the false negative rate; this can be mitigated by the following method.

2) Reducing False Negatives. To reduce the chance of false negatives, we can declare a Trojan if at least P out of Q chips exhibit leakage current past the rejection threshold, where P is defined earlier in 1) and $Q > P$. For a given P, the larger Q is, the fewer false negatives are expected.

A detailed analysis of tuning the parameters α, P and Q is beyond the scope of this paper, but a simple strategy may consist of the following four steps. 1) Choose α which gives approximately equal rates of both types of error. 2) Increase P until the desired false positive rate is reached. This may result in increased false negative rates. 3) Increase Q until the false negative rate is low enough. This may increase the false positive rate

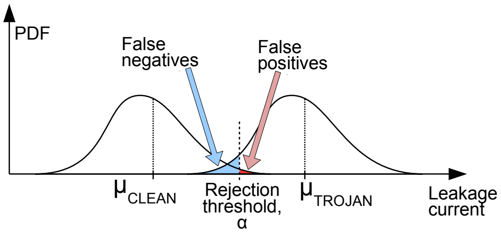


Figure 3: Diagram of probability distributions of total circuit leakage current for a clean and Trojan-infected circuit. The probability mass is shifted to the right by the magnitude of the added Trojan current.

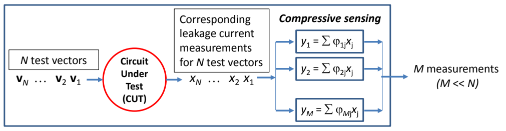

N test vectors
$\mathbf{v}_N \dots \mathbf{v}_2\ \mathbf{v}_1$
Circuit Under Test (CUT)
Corresponding leakage current measurements for N test vectors
$x_N \dots x_2\ x_1$
Compressive sensing
$y_1 = \sum \varphi_{1j} x_j$
$y_2 = \sum \varphi_{2j} x_j$
$y_M = \sum \varphi_{Mj} x_j$
M measurements (M << N)

Figure 4: The DISTROY front-end applies N randomly chosen test vectors to a CUT, measures corresponding leakage currents, and compresses to M linear combinations.

again. 4) Repeat steps 2 and 3 until both error rates are at acceptable levels.

4 DISTROY

DISTROY consists of the front-end scanner and back-end analyzer. In this section we describe these components of DISTROY in detail.

4.1 The Front-end

Figure 4 depicts the DISTROY front-end. The front-end applies N test vectors $v_1, v_2, \dots, v_N$ to a CUT, obtaining corresponding leakage current measurements $x_1, x_2, \dots, x_N$. We next use the compressive sensing matrix $\mathbf{\Phi}$ to reduce the measurements x_i down to M linear combinations y_j. Thus, instead of outputting N measurements from the chip, we now output only M measurements, with $M << N$. Unlike a typical data processing (e.g., compression) scheme that performs a significant amount of processing at acquisition, DISTROY handles the incoming data in a relatively light-weight manner by simply multiplying with $\mathbf{\Phi}$.

4.2 The Back-end

The back-end performs the decoding of compressive measurements y_i using the minimization of Equation (1). However, as noted in Section 2, to make decoding work with high probability, the variables under optimization must be K-sparse. But neither the expected measurements nor those of the CUT are sparse by themselves; how can we recover them using compressive sensing decoding?

Note that we are interested in finding the measurements which significantly deviate from the expected ones. Let us define a new set of variables, $d_1, d_2, \dots, d_N$ describing these deviations; more specifically, $d_i = x_i - g_i$. We can see that the deviations are going to be relatively more sparse; for example, in the ideal case without process variations, we would expect $d_i = 0$ unless a Trojan circuit is present. Normalizing by the standard deviation σ of leakage current, we can decode d_i using Equation (1)'s minimization as follows:

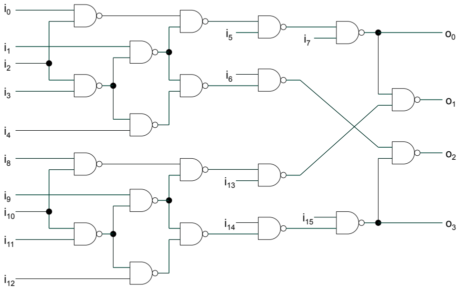


Figure 5: double-c17 combines two ISCAS-85 c17s.

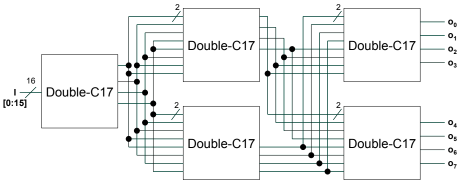


Figure 6: 100-NAND gate double-c17x5 benchmark circuit used for evaluation.

$$\min \sum \left| \frac{d_i}{\sigma_i} \right| \quad \text{subj.} \quad \mathbf{y} = \mathbf{\Phi} \begin{bmatrix} g_1 + d_1 \\ g_2 + d_2 \\ \dots \\ g_N + d_N \end{bmatrix} \tag{2}$$

The normalization is needed because of the "largest-first" decoding property of compressive sensing. Without the $1/\sigma_i$ factor in the objective function, the largest values we decoded might not be Trojan power consumption outliers, but merely largest power consumptions occuring in test vectors with high variance.

Having obtained the deviations d_i, we can now use the same types of statistical tests as in the baseline case of Section 3.

5 Evaluation

Our evaluation features a benchmark circuit that contains 100 NAND gates. We performed a logic simulation of the circuit and applied the Fernandes and Vemuri method [5] to model log-normal leakage current distributions. This section explains our evaluation methodology and discusses empirical results.

5.1 Benchmark Circuit and Trojans

The original c17 circuit from the ISCAS-85 benchmark suite [6] consists of 6 NAND gates; we combine two c17 blocks to create double-c17, which contains 20 NAND gates as depicted in Figure 5. Lastly, we use five double-c17 blocks to produce double-c17x5 shown in Figure 6.

The double-c17x5 circuit takes a 16-bit input, which yields a test vector space size of 2^{16}. Since this is a relatively small set, our simulation uses all possible test vectors and obtains corresponding gate states of the circuit for each vector. Furthermore, we compute the distribution of the sum of leakage currents through all gates in the circuit as described in Section 3.1.

Inserting Trojans. We prepared five unmodified double-c17x5s and placed one to five NAND gates at random locations to create trojan-1, ..., trojan-5. (That is, the smallest Trojan circuit is a single NAND gate.) We then ran logic simulations for the Trojan circuits and again used approximation to determine their leakage current distributions.

5.2 Performance Metrics

Achieved compression gain N/M. Enabled by compressive sensing, DISTROY can detect Trojans with M measurements for off-chip analysis that are several times fewer than the original N.

False positive rate. False positives occur when DISTROY pronounces a clean circuit Trojan-infected.

False negative rate. False negatives occur when DISTROY fails to detect a Trojan-infected circuit.

5.3 Trojan Detection Decision

We adopt the baseline detection method described in Section 3.2 and extend it to take advantage of multiple test vectors per chip, as well as multiple chips. First, for a single test vector, similar to Section 3.2, we declare the test vector a *Trojan witness* if its leakage current exceeds the mean by more than some threshold value 2σ.

Let us define an *equivalence class* of test vectors to be a set of test vectors that result in equal gate state counts. For example, if the test vectors v_3 and v_7 produce the state counts $\langle C_0 = 32, C_1 = 50, C_2 = 18\rangle$ where C_0 represents the total number of 2-NAND gates with input 00, C_1 with input 01 or 10, and C_2 with input 11, then the two test vectors are members of the same equivalence class.

We use equivalence classes as a convenient, easy to precompute tool to jointly reason about the statistics of multiple test vectors. Specifically, we apply the following criteria. We first divide the N test vectors used in a test into their equivalence classes. Then, we throw out all equivalence classes that have less than some number N_{Req} of member test vectors. Lastly, if we find that at least $L\%$ of test vectors in one of the remaining equivalence classes are Trojan witnesses, we declare the CUT Trojan-infected.

5.4 Discussion

DISTROY measures leakage currents induced by test vectors and compresses normalized deviations from the ideal values. We run $N = 1,000$ *random* test vectors on a clean, Trojan-free double-c17x5 and Trojan-infected double-c17x5s of five Trojan sizes, compress the current leakages with varying number of measurements (M), and count the number of Trojan witnesses.

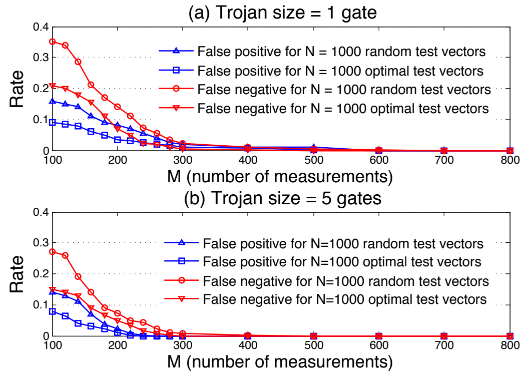


Figure 7: False positive and negative rates obtained from circuits containing Trojans of size 1 and 5 gates.

Figure 7 depicts the false positive and false negative rates for our smallest and largest Trojan circuits. We used parameters $N_{Req} = 20$ and $L = 50\%$. We have analyzed the leakage current distribution of all equivalence classes (using Fernandes and Vemuri [5]), ranked them, and selected test vectors from the lowest-power inducing equivalence classes to force the effect of additional gates from Trojan circuits more pronounced. We note that using some optimal set of test vectors we can reduce false positive and false negative rates.

We find that DISTROY can achieve up to *5- or 4-to-1* compression ratio, which justifies the use of compressive sensing for Trojan detection. We can think of compression as a speedup in the Trojan detection time by reducing the chip's output bandwidth requirement.

In Section 3.2.1, we discussed the use of multiple chips fabricated under the same process to improve reliability of the test. Figure 8 exhibits such an improvement. We consider the case for $M = 200$ (*i.e.*, the compression gain of 5). We ran 100 test cases using $P = 1$ to 10 chips from the same process and recorded the reduction of false positive rate. Using either random or optimal test vectors, we were able to achieve no false positives. The similar result holds true for false negative rate. Fixing $P = 2$ and varying Q from 2 to 10, we were able to eliminate false negatives starting with $Q = 6$.

6 Related Work

Agrawal *et al.* [1] introduced IC fingerprinting, a signal processing technique using *side-channel* power analysis to detect the presence of additional circuits. IC fingerprinting assumes a gold circuit fabricated physically

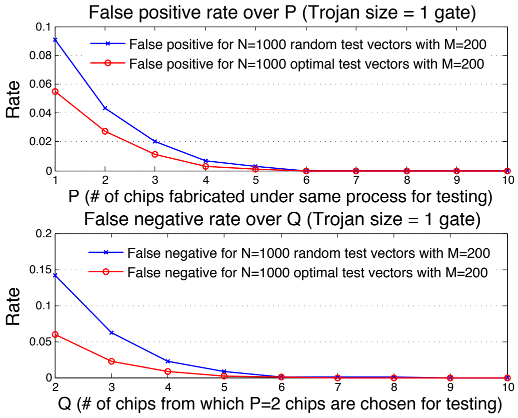


Figure 8: Using multiple chips fabricated under the same process improves false positive and negative rates.

to extract the reference fingerprint used in testing that serves decision criteria. The compressive sensing based approach offers a possible implementation direction for fingerprinting methods. Nelson *et al.* [9] presented gate-level characterization (GLC) techniques to model power and delay characteristics more precisely. GLC relies on statistical methods, singular value decomposition (SVD) in particular, to solve for a characterization vector used to detect a Trojan circuit. The compressive sensing approach of this paper can benefit from using such methods in selecting Trojan-revealing test vectors.

More recently, Trojan detection in ICs has attracted considerable research efforts at the *IEEE Symposium on Security & Privacy (Oakland)*. They include Huffmire *et al.* [8], an isolation primitive for hardware components to run on FPGAs that can help interconnect traceability among others but provides little protection against potentially malicious central component such as a Trojan-infected microcontroller chip. Hicks *et al.* [7] proposed Unused Circuit Identification (UCI) to detect malicious hardware hidden in circuits at design time by identifying pairs of dependent signals replaceable by a wire without affecting any test vector outcome. Sturton *et al.* [11] demonstrated a valid attack against UCI by showing that it is possible to build malicious circuits exhibiting hidden behavior upon receiving a special trigger and bypassing the UCI detection successfully.

All these approaches are orthogonal to the compressive sensing-based approach of this paper, which has the goal of reducing I/O requirements without compromising important information for detecting Trojans.

7 Conclusion

Trojans are a hard problem and serious security threat for today's fabless IC business model. We have presented DISTROY, a novel and unconventional use of compressive sensing to address the Trojan detection problem. Because of the *largest-first* decoding property, compressive sensing decodes the largest abnormal power consumption values first. From the decoded values, we can detect the Trojans reliably and accurately.

We have used a reasonable benchmark circuit for our evaluation and also for illustrative purposes. In the near future, we plan to validate DISTROY in implementation, applying it to real circuits.

Acknowledgments

This material is based on research sponsored by Air Force Research Laboratory under agreement numbers FA8750-10-2-0115 and FA8750-10-2-0180. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. The authors would like to thank the Office of the Secretary of Defense (OSD/ASD(R&E)/RD/IS&CS) for their guidance and support of this research. In addition, we'd like to thank our HotSec'11 reviewers for their helpful comments.

References

[1] AGRAWAL, D., BAKTIR, S., KARAKOYUNLU, D., ROHATGI, P., AND SUNAR, B. Trojan Detection Using IC Fingerprinting. In *IEEE Symposium on Security and Privacy* (2007).

[2] ALKABANI, Y., AND KOUSHANFAR, F. Consistency-based characterization for IC Trojan detection. In *Proc. of ICCAD* (2009).

[3] BARANIUK, R. G. Compressive Sensing. *Lecture Notes in IEEE Signal Processing Magazine vol. 24*, no. 4 (Jul. 2007).

[4] CANDÉS, E. J., AND WAKIN, M. B. An Introduction To Compressive Sampling. *IEEE Sig. Proc. Mag. 25*, 2 (2008), 21–30.

[5] FERNANDES, R., AND VEMURI, R. Accurate estimation of vector dependent leakage power in the presence of process variations. In *Proc. of ICCD* (2009).

[6] HANSEN, M. C., YALCIN, H., AND HAYES, J. P. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Design & Test 16* (July 1999), 72–80.

[7] HICKS, M., FINNICUM, M., KING, S. T., MARTIN, M. M. K., AND SMITH, J. M. Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically. In *IEEE Symposium on Security and Privacy (Oakland)* (2010).

[8] HUFFMIRE, T., BROTHERTON, B., WANG, G., SHERWOOD, T., KASTNER, R., LEVIN, T. E., NGUYEN, T. D., AND IRVINE, C. E. Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems. In *IEEE Symposium on Security and Privacy (Oakland)* (2007).

[9] NELSON, M., NAHAPETIAN, A., KOUSHANFAR, F., AND POTKONJAK, M. *SVD-Based Ghost Circuitry Detection*. Springer-Verlag, Berlin, Heidelberg, 2009, pp. 221–234.

[10] SINGH, A., GULATI, K., AND KHATRI, S. Minimum Leakage Vector Computation Using Weighted Partial MaxSAT. In *IEEE Midwest Symposium on Circuits and Systems* (2010).

[11] STURTON, C., HICKS, M., WAGNER, D., AND KING, S. T. Defeating UCI: Building Stealthy and Malicious Hardware. In *IEEE Symposium on Security and Privacy (Oakland)* (2011).

[12] WAKSMAN, A., AND SETHUMADHAVAN, S. Silencing Hardware Backdoors. In *IEEE Symposium on Security and Privacy (Oakland)* (2011).