Deterministic Circuit Variation for Anti-Tamper Applications

[Extended Abstract]

J. Todd McDonald
University of South Alabama
School of CIS
Mobile, AL 36688
jtmcdonald@usouthal.edu

Yong C. Kim
Air Force Institute of Technology
Dept. of Elec. and Comp. Eng.
Wright-Patterson AFB, OH 45433
ykim@afit.edu

Daniel Koranek
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson AFB, OH 45433
daniel.koranek@wpafb.af.mil

ABSTRACT
The electric power grid underlying our national infrastructure faces various challenges from adversaries that may exploit weaknesses gained through tampering and malicious reverse engineering. In this paper we describe a method for frustrating such adversaries based on polymorphic generation of circuit hardware with specific hiding properties in mind. We introduce component fusion as a technique for generating functionally equivalent variations of target logic that merge and blur the boundary between constituent components. We show how both random and deterministic variation can be combined to produce circuits that are efficient within allowable bounds while driving up cost of malicious tamper efforts.

Categories and Subject Descriptors
B.6.1 [**Hardware**]: Logic Design – *design styles*, *combinational logic*. K.6.5 [**Computer Milieux**]: Management of Computing and Information Systems – *security and protection, unauthorized access, invasive software.*

General Terms
Algorithms, Measurement, Design, Reliability, Security.

Keywords
Reverse engineering, anti-tamper, circuit protection, polymorphic generation, software protection, obfuscation, component identification, circuit variation.

1. INTRODUCTION
Computing technology remains the target of large investment for both the federal government and industry. Research and development efforts aim for state-of-the-art advancements, making technology a valuable commodity as well as the vehicle for handling our most valuable information. In terms of national infrastructure, older technology must give way to newer versions that will transition control and distribution of our national assets (including energy) for the next generation. Reverse engineering can shorten the technological advantage for any particular push forward that is made, whether commercially or as part of our national computing and power grids. Adversaries seek to understand technology in order to manipulate it or cause its compromise—these manipulations ultimately bear on national security interests themselves which translate to either critical systems failure or loss of life.

We seek to protect these valuable assets from such observation and manipulation using the art of the possible. Cryptography provides us apt protection for sensitive data, but analogous protections for physical hardware and computing logic are not as easily derived. Our interest concerns how to protect not just physical hardware, but general digital circuit definitions such as application-specific integrated circuits or field programmable gate arrays from malicious reverse engineering. *Obfuscation* of computing logic provides one possibility for degrading an adversary's capability.

We consider whether it is possible to construct efficient methods to generate securely obfuscated versions of combination logic. By security, we do not mean full protection of a circuit where no information leaks relative to the original circuit (i.e., a virtual black box [1]). Rather, we define security by the powers of an adversary to perform particular analysis tasks related to reverse engineering [2]. Section 2 gives further background on this definition. In developing algorithms to answer this question, three goals emerge: 1) variants must be semantically equivalent, computing the same output for all inputs; 2) variants should provide better security related to hiding structural and internal function information; 3) variants should be producible in a reasonable timeframe without limitation of resources and should themselves be significantly no larger than the original. With these goals in mind, we lay out the remainder of paper as follows: Section 2 defines terms for circuit variation and algorithms that produce variation, Section 3 presents our algorithm for producing anti-tamper security via component merging and Section 4 discusses results from initial experiments.

2. CIRCUIT VARIATION AS DEFENSE
Adversaries use reverse engineering to gain understanding of underlying systems through analysis of structure, dynamic operation, and functional observation—all for malicious purposes.

Obfuscation provides one form of protection to deter, degrade, or prevent adversarial actions along those lines.

For experimental purposes, we consider specifically the properties of programs expressed as combinational logic circuits. We model these circuits as Directed Acyclic Graphs (DAGs) and use traditional Boolean logic gates as vertices in the connected graph. Given a basis set $\Omega \in \{AND, OR, XOR, NAND, NOR, XNOR\}$, we define a circuit C as a DAG having either nodes mapping to functions in Ω (referred to as intermediate *gates*) or having nodes with in-degree 0 being termed *inputs*. *Outputs* are one or more distinguished intermediate nodes. Circuit behavior is representable as a Boolean function, $f:\{0,1\}^n \rightarrow \{0,1\}^m$, where n is the input size (or input length) and m is the output size (or output length) in bits. We refer to function-preserving changes in the underlying structure or graph of the circuit (replacing one or more gates with other Boolean logic gates) as white-box variation.

2.1 Defining Obfuscation

An algorithm, $O: \delta_\Omega \rightarrow \delta_\Omega$, is a polynomial time algorithm that takes as an argument circuit $C \in \delta_\Omega$ and returns a functionally equivalent version $C' \in \delta_\Omega$. For semantic equivalence of variants, it follows that $\forall x: C(x) = C'(x)$, where $x \in \{0,1\}^n$ for all possible $C \in \delta_\Omega$. O represents a white-box variation engine under this definition. We distinguish the nature of the obfuscating algorithm O and the distribution of circuits that it produces from some original circuit C.

On one hand, we desire O to be a publically known algorithm, consistent with Kerckhoff's principle[1], that makes pseudo-random mutations to produce a variant, where the particular choices made essentially forms a secret key. To maximize security of the obfuscator, we would prefer O to make a uniform, random choice from the subset of all possible circuits that have the same function as a candidate circuit C. Given this requirement, adversaries would gain no advantage from having the code for the algorithm O because the set of possible variants for any reasonably sized circuit is enormous. Under this definition, O would be equivalent in performance to the statistically indistinguishable and best possible obfuscators defined by Goldwasser and Rothblum [1]. Though efficient (tractable) obfuscators are not possible to produce under this definition, we consider whether we can approximate circuits with random, uniform properties through sequences of small random, uniform mutations.

On the other hand, we also desire the distribution of circuits (C') produced by O to demonstrate *security* properties of interest: in other words, any given C' variant should reveal less information in regards to one or more specific adversarial analysis vectors. These properties of interest include topological configuration, signal information, component configuration, control signal allocation, or other user-specific white-box characteristic [2,3]. We specify these particular characteristics because they lay the foundation for learning the abstract representation of any given circuit.

Whereas our first distinction defines the quality of the obfuscator itself, the latter distinction defines the properties of the circuit variants we want the obfuscator to produce. Our ongoing research to pursue both goals reveals that the two distinctions are not synonymous.

Initial research [3,4] reveals that attempts to *approach* a uniform, random distribution may not necessarily produce circuits with less information relative to a specific adversarial, white-box analysis goal. In addition, purely random, small-scale variation techniques at a structural level introduce redundant logic patterns that are reducible via application of Boolean logic laws [5]. Our initial hypothesis in [6] concerned whether approximating a truly random selection $C'_R \in \delta_C$ would produce variants that provide better security. Under a best possible or random program definition of statistical indistinguishability, no guarantee exists for variants to exhibit better security related to some original circuit. Our previous work in the area deals with designing obfuscating engines that use iterative, random white-box transformations and observing whether variant distributions approach uniform properties [4,5,6]. More importantly, we study whether variants produced by such obfuscators manifest hiding properties related to structural characteristics of a circuit (topology hiding, signal recovery, or in our current study, component hiding) which are essential for successful reverse engineering.

2.2 White-Box Variation

We define the domain set δ_Ω as the set of all possible circuits derivable from combinational logic based on basis (gate set) Ω. Within this infinite set of circuits δ_Ω, we can partition subsets based on circuits with n inputs and m outputs. We can narrow this partition further based on other circuit-related features such as gate size, number of levels, number of control paths, etc. For analyzing variants of a candidate circuit $C \in \delta_\Omega$, we use gate size and consider circuits that have the same gate size as C plus some allowable increase in size based on constraints of the environment (200% increase in gate size for example). We denote a circuit family $\delta_{n\text{-}m\text{-}S\text{-}\Omega}$ as the set of all circuits with input length n, output length m, maximum intermediate gate size S, and basis Ω. We let δ_C represent a subset of circuits that compute the same function, $f_C:\{0,1\}^n \rightarrow \{0,1\}^m$, as circuit C. Figure 1 depicts our domain of interest $\delta_C \subseteq \delta_{n\text{-}m\text{-}S\text{-}\Omega} \subseteq \delta_\Omega$ and operation of algorithm O.

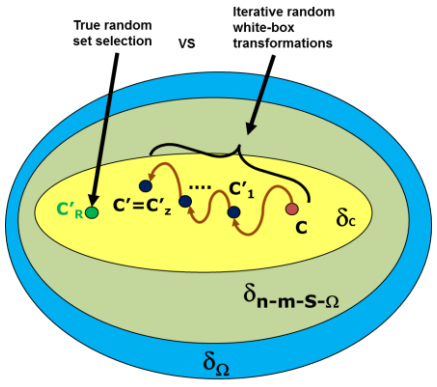

Figure 1. Domain of Interest for Circuit Variation

Figure 1 illustrates the concept of incremental white-box variation. Given a circuit $C \subseteq \delta_C \subseteq \delta_{n\text{-}m\text{-}S\text{-}\Omega} \subseteq \delta_\Omega$, a randomizing obfuscator that uses iterative function-preserving white-box changes will produce a sequence of variants: $\{C'_1, C'_2, \ldots, C'_z\}$, where for $j = 1 .. z$: $\forall x$, $C'j(x) = C(x)$. The final variant C'_z become the output of the obfuscator, $O(C) = C'$. The variable z represents some upper bound of iterations by the obfuscator and

[1] Auguste Kerckhoff in *La Cryptographie Militaire*: the security of a cryptosystem should rest solely in the secrecy of key and not the design of the system itself.

we use desired operational limits to constrain its value (maximum gate size, number of levels, etc.). All function preserving obfuscators generate and effectively choose a variant *C'* from δ_C. Figure 1 also illustrates how an ideal obfuscator that uniformly selects a replacement *C'* from the set of all possible elements in δ_C would produce a distribution so that $O(C) = C_1$ is statistically indistinguishable from $O(C) = C_2$. Likewise $O(C) = C_1$ is indistinguishable from $O(C_1) = C_2$. To accomplish this, an obfuscator in the ideal case would generate all possible variants in the set δ_C and then make a uniform selection, C'_R. For all but very small circuits (gate size <= 6), full set generation and random uniform selection remains intractable.

2.3 Random Variation vs. Achievable Hiding

Hansen et al. [6] list several reverse engineering techniques and adversarial goals: (known/standard) library modules, repeated modules, expected global structures, computed functions, control functions, bus structures, and common names. Because of the importance of components in building digital logic systems, we focus this paper on the adversarial goal of *component recovery* (or *module identification*)**:** the act of reproducing the architectural or component level relationships of the original circuit ([7,8]). Though other goals are important, component identification is by far the first and primary goal of a reverse engineer to organize lower level combinational logic into known abstractions which can be further studied.

3. Deterministic Variation

We define an obfuscating engine *O* that produces variation by using a sequence of small, incremental function preserving changes. The collected random choices of the *O* in making such choices form a secret key. Given the key, we can reproduce every individual change that went into producing a final variant. Without the key, we reduce an adversary to observing products of the obfuscating engine plus any information they derive from the obfuscator code itself.

We define our general obfuscating algorithm, $O(C) = C'$, as:

Given a circuit $C \subseteq \delta_\Omega$, let $C_0 = C$
FOR $i = 0$ to z:
1) SELECT a set of gates G_i as a subcircuit within C_i and let f_{Gi} represent its function
2) REPLACE subcircuit G_i with a version G_{i+1} such that $\forall x: f_{Gi}(x) = f_{Gi+1}(x)$
3) REMOVE subcircuit G_i from C_i and replace with G_{i+1}
4) let $C_{i+1} = C_i$

We consider each operation of the 'FOR' loop as an iteration. For each SELECT and REPLACE operation, we categorize the four possible strategies based on whether each select/replace operation is predominantly pre-determined or pseudo-random.

3.1 Random Selection and Replacement

Selection in our obfuscation algorithm is typically limited by the capabilities of the replacement engine itself. Our experimental work has focused on approaching an overall uniform random selection of the obfuscator by making uniform random selections during each iteration. In order to make a uniform random replacement, the size or number of gates in the selection sub-circuit has to remain small (size <= 6) because fully enumerating all circuits in the family $\delta_{n\text{-}m\text{-}S\text{-}\Omega}$ is intractable for larger sub-circuits. By using this approach, the obfuscator is limited in the possible variants reachable by random selection and replacement alone and we designate the set of these variants as $\delta_{C\text{-}REACHABLE}$. As figure 2 represents, we strive to find circuits that exhibit good hiding properties of interest, which we designate by circuits in the subset $\delta_{C\text{-}GOOD}$.

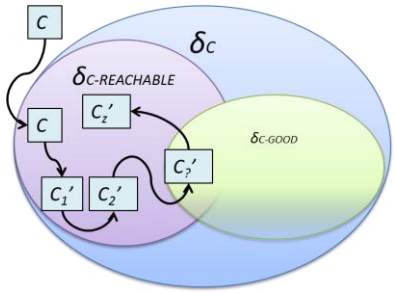

Figure 2. Limitations of Random Selection/Replacement

3.2 Deterministic Component Fusion

We define obfuscation research as the goal to find efficient algorithms that minimize the difference between the two sets: $\delta_{C\text{-}REACHABLE} \Delta \delta_{C\text{-}GOOD}$. In order to overcome limitations of random selection and random replacement, we define a deterministic selection and replacement scheme called *component fusion* to guide the variation process towards circuits within $\delta_{C\text{-}GOOD}$. We narrow the nebulous concept of which circuits exist in $\delta_{C\text{-}GOOD}$ by focusing only on circuit variants that exhibit component hiding properties. In previous results [5, 8], we have reported the general efficacy of deterministic strategies aimed at degrading adversarial component identification and identified circuits with certain component configurations which are not conducive to hiding with *any* obfuscation technique at all. Formally, given a circuit *C*, its gate set *G*, its input set *I*, and an integer $p > 1$, where *p* is the number of components, a set *M* of components $\{m_1,\ldots, m_p\}$ partitions *G* and *I* into *p* disjoint sets of inputs and/or gates. In component fusion, we modify the general selection/replacement algorithm defined earlier as follows:

Given a circuit $C \subseteq \delta_\Omega$ and let $C_0 = C$
Let G = the gate set of C and let $G_{UNUSED} = \emptyset$
Let $M = \{m_1, m_2, m_3, \ldots, m_p\}$, a component set of C
REPEAT
1) SELECT a component $m_i \in M$
2) PARTITION unused gates into connected subcircuits to produce component m_i'
3) MERGE component m_i' into C_i and add any changed gates to G_{UNUSED}
4) let $C_{i+1} = C_i$
UNTIL $G_{UNUSED} = G$

The SELECT and PARTITION operations of the algorithm use pseudo-random choices to drive the variation process so that each execution of *O* will produce a unique distribution of intermediate and final variants. The output of the PARTITION operation is itself a subcircuit that forms the basis for replacement which is passed on to the MERGE operation. The MERGE operation involves a deterministic approach defined as follows: 1) choose a random gate basis Ω; 2) choose a random Product of Sum/Sum of Products implementation as a canonical form; 3) using ESPRESSO's Quine McCluskey algorithm, logically reduce the component m_i' to generate a replacement subcircuit. Figure 3 depicts the overall operation of a given iteration in the algorithm. Figure 3-(1) represents the partition of *C* into component set *M*; Figure 3-(2) represents the SELECT of a component m_i plus the PARTITION operation which adds gates from predecessor components; Figure 3-(3) and 3-(4) depicts the MERGE operation which reduces the component m_i' back into the original circuit. This technique offers several advantages towards security

(defined as degradation of adversarial component recovery) and efficiency: 1) the component selection and Quine McCluskey reduction ensures replacement of the selected sub-circuit every iteration; 2) the use of component definitions hides known existing information specific to original component relationships; 3) predecessor addition ensures selection and replacement will always overlap; 4) deterministic replacement method increases speed of finding replacements while using random synthesis; 5) each replacement is a small version of a virtual black box.

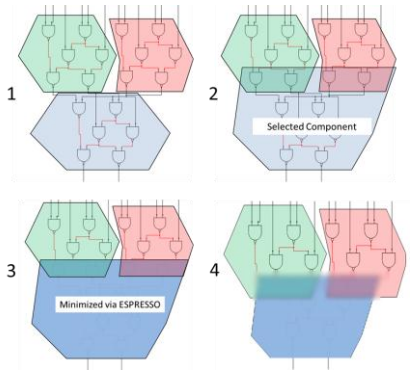

Figure 3. Component Fusion at a Glance

4. EXPERIMENTAL RESULTS

Initial results for experiments using the component fusion appear promising for both security and efficiency. We set up obfuscation experiments using the *c6288* 16-bit multiplier circuit that is part of the ISCAS-85 benchmark set. The *c6288* represents a good test case for component hiding because it is a 32 input/32 output circuit composed of 224 full-adder components and 16 half-adder components. Identification tools easily identify all components in *c6288* in a single pass within 2 minutes; we use the component identification algorithm of White as the basis for our adversarial recognition tool [9]. For comparison, we build on existing results from experiments using random selection and replacement (SSR) and component boundary blurring algorithms [8]. We execute the component fusion algorithm in 50 different experiments using the *c6288* as the candidate *C*. For comparison, we run the same experiments on algorithms based on SSR and boundary blurring and use component identification on the resulting variants. We represent the average number of components identified across variants as a percentage and show our results in Figure 4. In all cases, purely random selection and replacement allows identification of some or all of the original full-adders and half-adders from the original *c6288*. Component fusion improves component recovery results 37% over the best random selection/replacement technique and is comparable to the deterministic boundary blurring reported in other studies [8]. Figure 4 illustrates that some forms of SSR created minimal or no hiding of original components at all.

In conclusion, we report a small subset of our experimental results in figure 4, but note the following trends from our full study. Gate size in variants was on average 350% larger than the original circuit. Future work will aim to reduce the size of variants further using integrated logic reduction techniques. We observe results with other circuit families of interest and found similar trends regarding 100% component hiding using the identification tool. Additional work will seek to find other techniques for component identification for adversarial comparison. The value of the study indicates that completely unprotected versions of circuits provide no hindrance to a determined adversary. Deterministic variation techniques such as component fusion demonstrate empirically that adversaries cannot rely on component identification techniques to recover component information during the reverse engineering process, thus providing impetus for future study on a broader range of circuits.

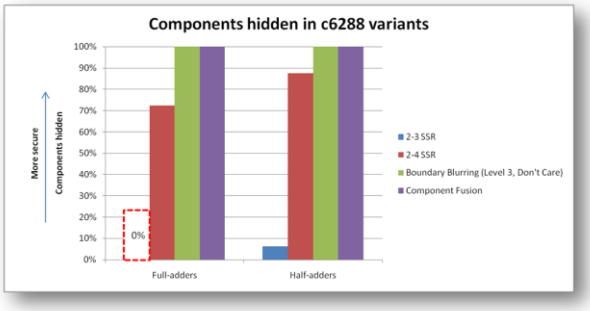

Figure 4. Component Identification under Component Fusion

5. ACKNOWLEDGMENTS

This material is based upon work supported in part by the U.S. Air Force Office of Scientific Research under grant number F1ATA09048G001. The views expressed in this article are those of the authors and do not reflect the official policy or position of the Unites States Air Force, Department of Defense, or the U.S. Government.

6. REFERENCES

[1] Goldwasser, S. and Rothblum, G. On best-possible obfuscation. *LNCS, Vol. 4392*, TCC 2007, Springer-Verlag, 2007, 194–213.

[2] Kim, Y. and McDonald J. Considering Software Protection for Embedded Systems. *Crosstalk*, 22, 6 (Sep/Oct 2009), 4-8.

[3] McDonald J., Kim Y., and Yasinsac A. Software issues in digital forensics. *ACM Operating Systems Review*, 42, 3 (April 2008).

[4] McDonald J., Kim Y., and Grimaila, M. Protecting Reprogrammable Hardware with Polymorphic Circuit Variation. In *Proc. of the 2nd Cyber Research Wrkshp* (June 2009), Shreveport, LA.

[5] McDonald J., Trias, E., *et al*. Using Logic-Based Reduction for Adversarial Component Recovery. In *Proc. of the 25th ACM SAC* (March 2010), Sierre, Switzerland,

[6] Yasinsac, A. and McDonald, J. Of unicorns and random programs. In *Proc. of 3rd IASTED CCN (2005)*, Marina, CA.

[7] Hansen, M., Yalcin, H., and Hayes, J. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *IEEE Design & Test of Computers*, 16, 3 (1999), 72–80.

[8] Parham J., Kim Y., *et al*. Hiding Circuit Components Using Boundary Blurring Techniques. In *Proc. of IEEE Annual Symposium on VLSI* (Jul. 5-7, 2010), Cephalonia, Greece.

[9] White J., Wojcik, A., *et al.*, Candidate sub-circuits for functional module identification in logic circuits," *Proc. of the 10th Great Lakes Symposium on VLSI* (2000), Chicago, IL, 34 – 38.