Protecting Integrated Circuits from Piracy with Test-aware Logic Locking

Stephen M. Plaza
Janelia Farm Research Campus, HHMI
19700 Helix Drive
Ashburn, VA 20147
plazas@janelia.hhmi.org

Igor L. Markov
EECS Department, University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109
imarkov@eecs.umich.edu

ABSTRACT

The increasing IC manufacturing cost encourages a business model where design houses outsource IC fabrication to remote foundries. Despite cost savings, this model exposes design houses to IC piracy as remote foundries can manufacture in excess to sell on the black market. Recent efforts in digital hardware security aim to thwart piracy by using XOR-based chip locking, cryptography, and active metering. To counter direct attacks and lower the exposure of unlocked circuits to the foundry, we introduce a multiplexor-based locking strategy that preserves test response allowing IC testing by an untrusted party before activation. We demonstrate a simple yet effective attack against a locked circuit that does not preserve test response, and validate the effectiveness of our locking strategy on IWLS 2005 benchmarks.

1. INTRODUCTION

The challenges of counterfeit electronics have been publicized by US Congressional hearings in November 2011 after large quantities of substandard devices were found in military electronics. They were featured on CNN and covered in depth by a US Department of Commerce study [5]. To this end, we seek to counteract *unauthorized manufacturing* (overbuilding) and support *active-metering techniques*. As explained in [8], the heavy costs of semiconductor manufacturing force IC supply chains to stretch across the Pacific. By outsourcing fabrication, the owner of IP rights invites unauthorized production of black-market ICs that undercut legitimate ICs. Whether such *acts of IC piracy* are condoned by the management of fabrication facility, or committed at a different facility, does not diminish the damage caused and does not help address the problem. Another threat model is to intentionally produce degraded or altered ICs looking like normal ICs, so as to facilitate sabotage. Despite significant interest from US DoD and Legislature, these challenges have been ignored by the EDA industry until recently. Yet, in the late 2013, major EDA companies started exploring business opportunities in the fight against IC piracy.

Threat models in IC piracy and countermeasures are reviewed in [18]. A key countermeasure developed in [1, 2, 7, 8, 19] is *active IC metering*, which forces every new IC produced to be activated through real-time electronic contact with owners of IP rights. Otherwise, the IC will not work. Active-metering schemes [1] have recently attracted significant interest: in addition to suggesting improvements, [13] noted a step-ordering ambiguity in the DATE 2008 version of EPIC [19], which was clarified in the journal version [19]. Exploits for the DATE 2008 version of [19] were also claimed by [16], offering ways to strengthen the proposed protocols.

We focus on a key feature of EPIC [19] — the need to activate the chip before circuit test, most likely at the fabrication facility. This unnecessarily exposes the activation protocol and the unlocked ICs, facilitating various attacks, as shown in [13,19]. To reinforce this point, we develop a new algorithmic attack that uses test-patterns and observed responses. The algorithm performs a randomized local search with restarts, guided by the number of matching bits in the output response, that typically produces a correct key value. The (surprising) empirical success of this key-extraction attack hints at an underlying mathematical structure in the behavior of large combinational circuits.

To thwart the new attack and to rule out attacks suggested in [13,16], we develop a *methodology for combinational locking that supports post-manufacturing test of locked circuits before activation.* Unlike prior methods that insert `XOR` gates (Figure 1), our new combinational locking inserts multiplexors. The insertion is based on functional simulation and tries to match *logic covers* of internal signals. The importance of performing IC testing before IC activation *as a security measure* was recently articulated in [3]. Their strategy requires significant infrastructure for remote testing and on-chip logic to scramble test response, including cryptography and binary tags. In comparison, our proposal is lightweight. The remaining part of this paper is structured as follows. Section 2 outlines relevant background on active IC metering, highlighting its key aspects considered in our research, then reviews recent literature and interaction with circuit test. It also articulates technical opportunities pursued in this paper. Section 3 introduces an algorithmic test-based attack on EPIC and partially motivates the development in Section 4 of combinational locking to enable post-manufacture test of locked circuits. Empirical validation on IWLS 2005 benchmarks is reported in Section 5, which also discusses embedding of proposed techniques in realistic design flows. Conclusions are given in Section 6. Readers confused by inconsistent terminology in recent literature may benefit from the Appendix.

978-1-4799-6278-5/14/$31.00 ©2014 IEEE

2. BACKGROUND: ACTIVE METERING

Given recent interest in active metering [1], we illustrate it by combinational locking in the EPIC protocol [19], and review its cost-vulnerability tradeoffs (a formal description of EPIC can be found in [13], along with analysis and improvements). We also briefly discuss attacks from [16].

EPIC. Building on prior work [1], [19] proposes a chip-locking and activation system for IC metering, while aiming to make "physical tampering unprofitable and attacks computationally infeasible." In other words, defending against an omnipotent attacker with unbounded resources is not the goal — simpler attacks should be given priority.[1] For example, reverse-engineering (a part of) circuit layout is harder than running a live circuit on given inputs. Observing internal dynamic voltage levels in a live circuit is more difficult than observing static gates or wires. Modifying a circuit typically requires understanding some part of it. Since EPIC draws on unique process variations to ensure different responses in ICs produced from the same mask, an attack that requires work for each individual IC will require considerable resources. Due to low margins in the IC business, a per-chip cost increase can make mass-production unprofitable. Moreover, physical inspection and modification of ICs are becoming increasingly challenging at each new technology node due to smaller features.

To establish a combinational lock, EPIC [19] modifies a combinational circuit by adding `XOR`/`XNOR` gates with fanins connected to the bits of common key (CK) that unlocks the circuit, as shown in Figure 1. Correct key bits simplify the `XOR`/`XNOR` gates to wires, while incorrect key bits produce unintended inversions. Locked IC will fail post-manufacturing test, hence unlocking must occur at the fabrication facility before test. Care is taken to avoid circuit delay overhead on critical paths [19]. Simple removal of `XOR`/`XNOR` gates is ineffective if inversions are propagated through the circuit and/or logic restructuring is performed after locking. The work in [2] is similar in principle to EPIC but uses LUT-based locks that hinder attempts to reverse-engineer functionality from the layout.

Several attack vectors. Both EPIC [19] and the analysis in [13] contemplate sophisticated attackers that obtained CK with some effort (guessing CK is shown difficult in [19]). They note that *EPIC does not provision for direct entry of CK to unlock the circuit.* Instead, the encrypted version of CK arriving from the owner of IP rights is protected by the RSA cryptosystem, which offers strong guarantees both in theory and in practice, though implementation-specific vulnerabilities exist [15]. The work in [16] assumes that the circuit has been reverse-engineered and develops attacks that simplify the search for CK, using unfortunate configurations of locking gates that may be created when inserting gates at random. Some of these configurations can be optimized during logic synthesis, and others are easy to avoid, e.g., as suggested in [16]. More critically, the authors of [16] consider an attack successful when CK is found and then focus on hiding CK better. In this context, recall that (*i*) reverse-engineering a large 22nm IC is going to be extremely difficult without access to the gate-level netlist, (*ii*) even if CK is found, entering it directly through mask modification demands advanced expertise and access to adequate infrastructure, increasing attacker's costs and barriers. In comparison, attacks discussed in [13] can be perpetrated by intercepting communications during activation session.

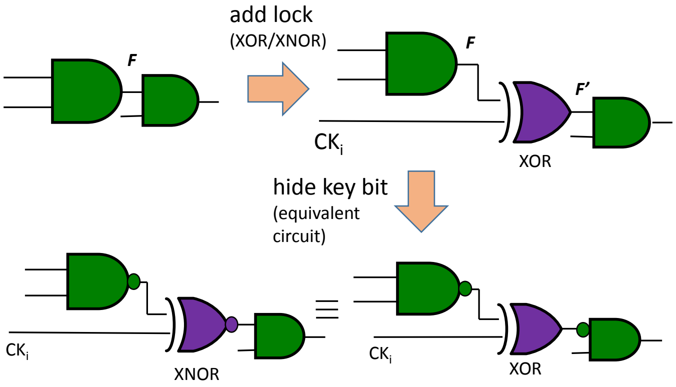

Figure 1: Locking a combinational circuit with XOR/XNOR gates. Initially, the key bit is represented with either an XOR or an XNOR gate, but then hidden by propagating inversions upstream or downstream in the circuit, using Boolean equivalences and subsequent logic optimization.

In Section 3, we describe a surprisingly effective CK-finding attack that uses either (*i*) a reverse-engineered or stolen gate-level circuit for simulation, or (*ii*) a live circuit with controllable inputs and accessible outputs. It does not make assumptions about how gates are inserted and is therefore completely unaffected by logic optimization or restructuring. The conclusion one should draw from this attack, as well as from those in [13, 16], is that *chip activation should not be performed at the fabrication facility.* On the other hand, as EPIC combines multiple layers of protection, viewing it as either secure or defeated would be misleading.

Observations and opportunities. Returning to the algorithmic description of EPIC, we note its interaction with circuit test. Activation is performed at the fabrication facility before circuit test because locked ICs will fail test. This exposes additional information to the manufacturer and enables additional attack vectors.

- The unlocked functionality is known after IC activation and during test, which may simplify and accelerate search for CK, including its encrypted form that can be entered directly,
- Including CK bits in post-manufacturing scan-test of locked circuits can significantly undermine EPIC if this facilitates entering CK bits directly [4, 17]

To improve IC security, testing and fabrication can be performed at separate locations by different commercial and legal entities. Such an arrangement can be described as *split test.* It has been explored in [3], at the cost of significant overhead. We pursue split test in a context that avoids significant changes in infrastructure by relying on more sophisticated ATPG algorithms. For further security, tamper-resistant packaging can deny access to activated chips available on the open market. On the other hand, ICs intended for sensitive military equipment (commonly mentioned as

[1] There is hardly an effective defence against an attacker who can reverse-engineer, replicate, and alter any IC design.

motivators for IC security research) should not be available on the open market.

Split test requires support in ATPG algorithms. To this end, we develop techniques to find patterns for testing both locked and unlocked ICs, so as to move activation from the fabrication facility without hampering yield optimization. Given the multiple concerns addressed by modern circuit test and its logistical complexity, it is important to maintain significant freedom for test-patterns.

3. AN EPIC ATTACK

We now introduce an algorithmic attack on EPIC that derives CK by simulating a set of test patterns. It can be executed on a stolen or reverse-engineered gate-level netlist, or on a physical circuit with direct access to CK inputs and circuit outputs (e.g., through scan-chains or maliciously inserted side-channels). Without the gate-level circuit, the attack additionally needs (*i*) test patterns and expected output responses, or (*ii*) an unlocked circuit that can produce correct responses on random inputs.[2] On the other hand, since EPIC does not offer direct control of CK bits on an actual IC (Section 2), using the results of the proposed attack would require malicious, although small, mask modification.

```
Input: Locked Circuit: ckt, Patterns: patterns
Response: resp
Output: CK: kbits {0,1}
found = false;
while found do
    kbits ← list random{0,1};
    foreach kbit in kbits.random() do
        resp1 = test(ckt(kbits), patterns);
        diffs1 = diff(resp1,resp);
        // flip one random bit
        kbit = !kbit;
        resp2 = test(ckt(kbits), patterns);
        diffs = diff(resp2, resp);
        if diffs1 < diffs then
            // original value of random bit
            kbit = !kbit;
            diffs = diffs1;
        end
        if !diffs then
            if correct(ckt) then
                found = true;
            end
            break
        end
    end
end
```

Algorithm 1: Extracting CK bit values from a locked circuit given test input and output vectors using a hill-climbing algorithm that monitors output response.

Our attack, introduced in Algorithm 1, is iterative in nature: a random key candidate is gradually improved based on observed test responses. It uses hill-climbing search guided by the number of differences in the output response (for a key combination). At each iteration, randomly-selected key bits are toggled one by one. The function `test()` applies all test patterns to the current key combination. A key bit value of 0 or 1 is chosen to minimize these differences. As described so far, the algorithm resets all key bits if no solution is found in one iteration. In practice, this `foreach` loop can be run multiple times until a local minimum is reached, followed by a restart with a new random configuration. The function `correct()` is an oracle that checks whether the current key combination, which satisfies the test response, is equal to CK. In practice, the oracle could be implemented through more exhaustive validation with an unlocked circuit or other expected response. If `correct()` is called several times, we can save lock-down key bits (not shown in algorithm) that do not vary between each *solution*.

Unlike the work in [16], our proposed approach does not require isolated sensitization of key bits or even netlist access (assuming that the scan chain is exposed). The insight is that the output response often betrays a *gradient* toward the target configuration (this need not occur every time, but sufficiently often). A key combination with fewer errors indicates an improvement in key combination. To develop intuition, consider a circuit where each XOR lock impacts a circuit output not impacted by other XOR locks. For a random combination of key bits, the output will differ from the expected test response by M bits. Toggling one key bit produces a different M value, and of the two resulting key bit combinations, the one with fewer differences is closer to a correct combination. Repeating such steps from a random initial combination often leads to a combination that unlocks the circuit. If not, the process can be restarted from a different random initial combination. The same logic applies with random placement of XOR locks, as in [19], as well as alternative techniques.

The complexity of this algorithm, in terms of the number of iterations required, does not directly depend on the size of the circuit. A higher density of locks in the circuit could add complex correlations [16] and jeopardize our strategy. However, more sophisticated strategies may explicitly seek key bits that impact the output in a correlated way, involve randomization, and maintain a pool of candidate configurations not to get trapped in local minima. More advanced attacks would examine the circuit's response to constrained stimuli. Among countermeasures against gradient-based attacks, we mention mapping CK key bits to pseudorandom locking combinations.

4. TEST-AWARE LOCKING

In this section, we introduce a strategy for locking a circuit that preserves test responses and is therefore immune to our proposed attack, as its response betrays no gradient. We advocate a lightweight locking approach (where the CK is encrypted) that allows a manufacturer to test the chip without a fully functional circuit. While our locking strategy does not rule out attacks that consider any circuit output response in the spirit of Algorithm 1, it significantly complicates such efforts as test response is identical for all key combinations. An attack would require comparing outputs with an unlocked chip (or netlist) and potentially a prohibitive simulation to expose deep circuit state.

We first introduce an approach using *logic signatures* to find

[2] Security pitfalls of scan-chains are well-known [17] and are being addressed through compression and encryption [4].

logic covers that preserve test response. Then we introduce an algorithm for inserting combinational locks in a circuit.

4.1 Test-Proof Locking using Logic Signatures

In [19], a lock at node F consists of creating an alternative signal F' created by adding an XOR (or XNOR) between F and a key bit. An incorrect key bit value gives $F' \equiv \neg F$; the correct one gives $F' \equiv F$. $\neg F$ will not preserve the test response unless F is redundant. Alternatively, we try to find (or synthesize) an F' that preserves the test response as explained in the next few paragraphs.

A *logic signature* is a partial truth table that captures the function of a given circuit node [10]. For a circuit and its K input vectors $X_1 ... X_K$, the logic signature of a functional node F in the circuit is:

$$S_F = \{F(X_1), ..., F(X_K)\} \tag{1}$$

Evaluating all input combinations turns S_F into a complete truth table. In practice, a set of random input vectors applied to a circuit can provide a useful mechanism for analyzing restructuring opportunities, such as identifying potential node equivalences [11, 14]. Since $S_F = S_G$ does not imply that $F = G$, such equivalences must be verified in general (more on this below). The time complexity of producing K-bit signatures for an N-node circuit is $O(NK)$. Signatures are generated quickly, as K is typically small. We generate signatures using both random input vectors and test input vectors. Figure 2a illustrates a circuit stimulated with different test vectors. For instance, $S_{X_4} = \{0, 1, 1\}$. Given these signatures, we seek an alternative implementation of a signal (in this example x_3) that preserves test response. We can look first for a node in the circuit that has an equivalent signature to S_{x_3}. Since no such signature exists, we must synthesize a function. We use the signatures to identify nodes that *cover* x_3 (similar to the strategy in [9]). A logic cover Y of x_3 is defined as:

$$S_{x_3} \subseteq S_Y \;\Rightarrow\; S_{x_3} \;\&\; S_Y \equiv S_{x_3} \tag{2}$$

(we say that Y covers x_3 up to given test vectors). In Figure 2a, signature S_{x_4} covers signature S_{x_3} since $x_4 = 1$ every time $x_3 = 1$ for the input patterns that define the signatures.

Unlike previous work, we do not need to formally validate that a candidate logic cover or node equivalence exists for all possible input combinations. On the contrary, we must show that input combinations that are not test vectors violate the equivalences. Figure 2b shows that we can use random simulation to generate another set of signatures that disproves the logic covers previously found. In this example, S_{x_4} does not cover S_{x_3}. Therefore, x_4 does not cover x_3. Any function that replaces x_3 with $x_3 \& x_4$ will preserve test response but will alter the circuit's behavior in general. Figure 2c shows that a `MUX` gate can be added where the select is the locking key bit that chooses between the correct x_4 and $x_3 \& x_4$.

4.2 An Algorithm for Adding Key Locks

We outline our approach to circuit locking in Algorithm 2, which assumes the circuit to be locked and test patterns as inputs. It first generates two sets of signatures, one generated from random simulation and one from test-pattern simulation. The algorithm randomly traverses the netlist trying to find a logic cover for the selected signal. `iscover()` is true if a cover is found but disproved by random simulation. If there is a cover, the function `cover()` synthesizes it from S_1 and S_2 (or returns S_2 if it is equivalent to S_1). To ensure that the differences between S_N and S_1 propagate to the output, we check its observability under random simulation. `insertmux` replaces S_1 with the output of a `MUX` between S_1 and S_N. The algorithm terminates once *numlock* locks are added.

```
Input: Circuit: ckt, Patterns: patterns,
Number of locks: numlock
Output: Locked Circuit: ckt
randsigs = simulate(ckt);
testsigs = simulate(ckt, patterns);
foreach $S_1$ in random({ckt.signals}) do
    foreach $S_2$ in ckt.signals do
        if iscover($S_1$, $S_2$) then
            $S_N$ = cover($S_1$, $S_2$);
            if ckt.observable($S_N$) then
                newkey = random({0,1});
                if newkey then
                    ckt.insertmux($S_1$, $S_1$, $S_N$);
                else
                    ckt.insertmux($S_1$, $S_N$, $S_1$);
                end
                numlock–;
                break;
            end
        end
    end
    if !numlock then
        break;
    end
end
```

Algorithm 2: Inserts MUX locks in a circuit where key bits choose between the correct signal and a logic cover.

In this approach, `cover()` generates S_N as a function of S_1. In general, any signal or combination of signals equal to S_1 under test simulation could be used. However, using S_1 is advantageous since it may be testable even with the wrong key bit value. Also, S_1 and S_N would ideally be local to reduce violations of aggressive design constraints.

To guarantee that an incorrect key input leads to a difference at the outputs, we check the observability of each locked signal `ckt.observable` . Hence, incorrect key bits result in a malfunctioning circuit. Our experiments check the observability of a locked signal with a set of random input patterns assuming that this is a subset of expected circuit payload (and that corresponding output values represent valid state). In more realistic settings, simulation patterns can be chosen among valid input states; output differences can be checked against valid output states. Two locked signals could theoretically cancel each other out, but even if such cases are not explicitly ruled out when positioning locks, this is astronomically improbable when the number of inserted MUXes (64 or 128) is small compared to circuit size.

A locked signal can be untestable given a wrong key bit value. In a circuit with N potentially untestable locked sig-

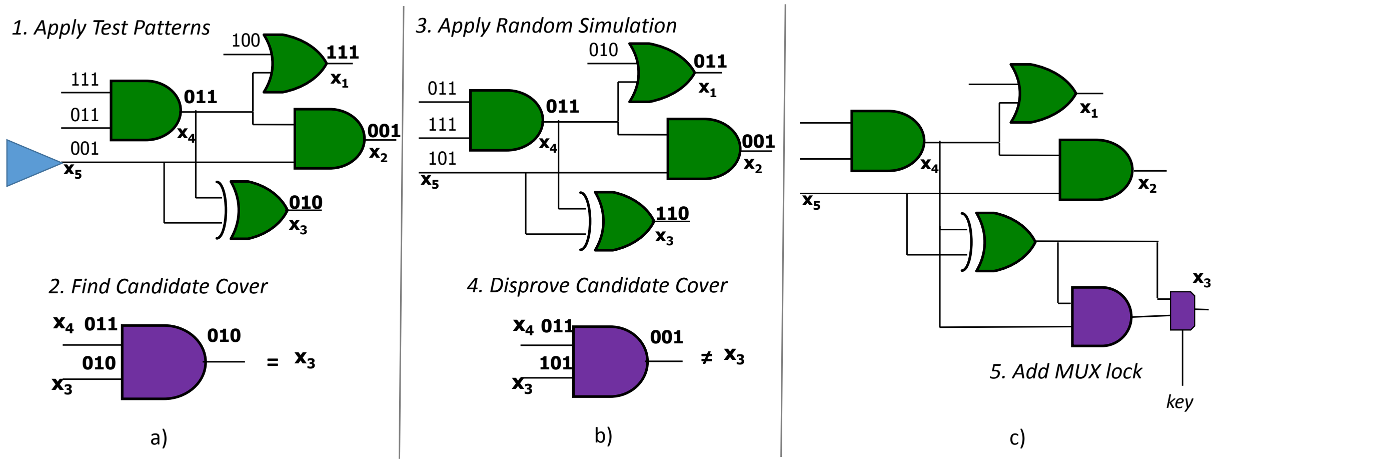

Figure 2: Signatures are used to add locks to a circuit so as to preserve circuit-test properties: (a) signatures can identify potential covers, (b) random simulation disproves the cover, (c) the insertion of a MUX between the logic cover and the locked signal that preserves the test vectors but are different under random simulation.

nals, we can test them by repeatedly applying test patterns with different key combinations. If there is little interference between locked circuit regions, the number of untested signals for t testing iterations is roughly governed by:

$$N_{untested} \approx \frac{N}{2^t} \qquad (3)$$

Notably, by ensuring that added locks are observable under random simulation, we miss potential locks that are rarely stimulated, which could enhance the resistance to simulation-based attacks. Also, we could make our approach visually harder to decipher (for an attacker with netlist access) by emphasizing locks that use local signals that are equivalent up to test vectors. Extracting the unlocking key bit value would be difficult since it is determined by the order of the MUX inputs.

5. EMPIRICAL VALIDATION

Our experiments use circuits from the ISCAS89 and IWLS'05 [22] benchmark suites. We derive combinational circuits from sequential ones by removing sequential elements and exposing their inputs and outputs to the whole circuit. Test patterns are generated using ATALANTA [12] with default settings. IWLS benchmarks are structurally hashed using ABC [21]. Circuit simulation, lock insertion, and key combination attack algorithms are implemented in C++, compiled by g++4.6, and run on an Intel Core i7 workstation.

5.1 Attacks using Logic Simulation

We now demonstrate that locking strategies that impact test response are susceptible to the attack in Algorithm 1. In particular, circuits that use XOR-based locking schemes can be unlocked by examining the gradient of output response.

As in [19], we randomly insert 32, 64, and 128 XOR key gates in each circuit (this is not a limitation of the experiment, but rather a representative strategy). Table 1 reports results for our attempts to determine CK from the provided test input and output response, as introduced in Algorithm 1. The results are averaged over 4 runs to produce detailed results, although in practice it is sufficient for a single run to succeed. The `keys` column indicates the number of different key combinations tried within our limits (1,000,000 combinations or two hours of simulation). For each key combination, `test vecs` is the number of test patterns applied.

We successfully unlock all circuits with 32 XORs, typically requiring only around 1000 key combinations. For circuits with 64 locks, we correctly determine the keys for all circuits except `c880` and one run of `usb_phy`, as indicated by the column `%Extracted`. These circuits are more resistant to attacks due to their small size and the relatively large number of locking gates added. As outlined in Section 3, the number of iterations in Algorithm 1 does not grow with the size of the circuit, but rather depends on the interactions between different locked signals. Also, the key combinations needed for `usb_phy` vary greatly as one run produced a circuit that was more resistant to attacks. Potentially, a more sophisticated simulation-based search strategy (rather than randomized hillclimbing) could better isolate interactions between different XOR key bits. When using 128 XOR locks, the interactions between them increase and hinder ability to discern the key combination. Despite this, we unlock `spi` in three of our trials. Furthermore, gradient descent quickly determines the majority of key bits for all circuits.

It is possible to find key combinations that satisfy the output response but do not unlock the circuit. This happens in over half of the circuits, as indicated by `False Match`. In other words, multiple global minima may exist. For a given circuit, if our gradient descent algorithm does not unlock the circuit after 10 random restarts, we analyze the common key bits between these key combinations (the randomness in our algorithm results in finding different minima). By removing the key bits common between these combinations, we effectively reduce the number of key bits to be examined. Then we re-solve the resulting smaller problem instance. As might be expected, larger circuits tend to have more global minima. Decreasing the observability of the locks increases the number of combinations that produce equal test response.

Table 1: Extracting a 32, 64, and 128 bit key using test patterns. The number of test patterns applied to each key combination is given by VECS. KEY indicates the number of keys tried. FALSE MATCH indicates the number of times a key was found that preserved test results w/o unlocking the circuit. %EXTRACTED is the percentage of random runs where the circuit was unlocked.

CIRCUIT	GATES	TEST	32 bit			64 bit			128 bit		
		VECS	KEYS	FALSE	% EX-	KEYS	FALSE	% EX-	KEYS	FALSE	% EX-
				MATCH	TRACTED		MATCH	TRACTED		MATCH	TRACTED
c880	383	53	65560	2	100	-	0	0	-	0	0
usb_phy	1197	67	1739	2	100	37709	1	75	-	0	0
sasc	1651	67	42	0	100	1334	2	100	174517	2	25
c3540	1669	149	1274	1	100	22624	2	100	-	0	0
i2c	2902	164	1061	4	100	11722	0	100	-	1	0
pci_spoci_ctrl	3483	246	839	32	100	13196	98	100	-	0	0
systemcdes	9008	123	231	1	100	4767	53	100	55005	205	50
spi	**10109**	**554**	**605**	**25**	**100**	**2290**	**57**	**100**	**13112**	**606**	**75**
tv80	22575	878	333	4	100	1727	42	75	-	1	0
systemcaes	26717	426	91	1	100	2247	6	100	489	4	50

While our attack succeeds surprisingly often, less robust attacks may also cause heavy damage, especially when they can be repeated until success.

5.2 Circuit Locking with MUXes

In this section, we validate the effectiveness of locking a circuit with MUXes so as to preserve test response. We demonstrate that there are several such transformations available in a circuit, and that, in general, these locks do not undermine fault diagnosability or create significant area overhead.

Table 2 shows the results of randomly adding 64 logic locks using MUXes and logic covers. For the more realistic larger circuits, the gate area increase (shown under %OVERHEAD) was minimal. Unlike [3], we do not require additional logic to lock each scan chain. All circuits contain several locking opportunities (as indicated in %CANDIDATE). This column denotes the percentage of wires with at least one cover. As a first-order strategy to preserve path timing, we skip logic covers that increase the number of logic levels in the circuit. As noted previously, additional flexibility in choosing logic covers can be attained by considering covers without using the locked signal. Finding locking candidates by simulation is fast, as reported under `time(s)`. The larger runtimes in `spi` and `tv80` are due to several potential locking sites being unobservable at the circuit outputs under a small set of random simulation patterns.

Table 2: Locking a circuit with a 64-bit key by finding logic covers that preserve testing but scramble the output response. %OVERHEAD is the increase in circuit size. %CANDIDATE is the percentage of wires that have a candidate cover. TIME(S) gives runtime.

CIRCUIT	%OVERHEAD	%CANDIDATE	TIME(S)
c880	33.42	66.39	1.81
usb_phy	10.69	54.01	1.03
sasc	7.75	57.48	0.19
c3540	7.61	58.47	116.94
i2c	4.41	52.99	4.74
pci_spoci_ctrl	3.67	73.36	12.28
systemcdes	1.42	66.97	6.94
spi	1.23	82.60	499.19
tv80	0.56	85.53	506.06
systemcaes	0.48	34.33	32.76

The addition of locking logic could limit the testability of the circuit. We briefly explore the coverage of gate output stuck-at faults for the MUX-locked designs (sans the locking logic, which is small). The first two columns of Table 3 show the fault coverage `%COV` and the number of untested faults `#UNTEST` in an unlocked circuit. The next two columns show results for the corresponding locked circuit with a random key combination. The coverage is nearly identical. This is due to (*i*) the number of locked sites being a small fraction of the design and (*ii*) synthesizing logic covers that include the locked logic such that it is still exposed to fault coverage. Notice that in some cases, like `spi`, the fault coverage actually improves. The added MUX logic can produce more sensitizing paths in the design, resulting in higher coverage.

Table 3 illustrates how the output response of a locked circuit compares to that of an unlocked circuit when a fault occurs. To aid circuit diagnosis, it is desirable for the locked circuit's output response to match the unlocked circuit. While our locking algorithm does not explicitly consider the impact of locking on diagnosability, we observe that most locked circuits achieve equivalent output response for about 90% of the faults with a random key combination. Note that `c880`, the smallest circuit, only matches 60% of the faults. To improve diagnosability, the same test patterns can be applied with different key combinations, producing matching output response for at least one of the combinations. Five testing runs of `c880` result in almost 90% of faults matching. The high percentage of matches with our locking strategy is due to (*i*) a relatively small amount of added locking logic and (*ii*) the locking strategy aiming to not change behavior under test in a functioning circuit.

5.3 Use in Industrial Design-and-Test Flows

Industrial IC design flows are remarkable in their handling of multiple optimization objectives and constraints. When incorporating a new technique, there is danger of disturbing carefully optimized tradeoffs and the overall stability of the design process. In the context of circuit test, it is important

Table 3: Fault coverage of a circuit with test-aware locking (using gate output stuck-ats). The first two columns show coverage (%COV) and untested faults (#UNTEST) in unlocked circuits. The next two columns show coverage in locked circuits for random key assignments. The final five columns show how the output response of a locked circuit compares to an unlocked circuit in the presence of single faults in a circuit. A high percentage indicates that a high number of faults produce the same output in both locked and unlocked circuits. Checking the output response for different random key combinations increases the percentage of faults that have output response that matches an unlocked circuit.

	Unlocked circuits		Locked circuits		% identical output response				
					(function of # random combinations)				
CIRCUIT	%COV	#UNTEST	%COV	#UNTEST	1	2	3	4	5
c880	100.0	0	99.5798	3	60.36	70.87	77.03	87.11	87.25
usb_phy	95.31	103	95.2641	104	86.38	92.30	92.71	95.99	96.58
sasc	98.65	41	98.458	47	91.63	96.10	97.54	98.16	98.46
c3540	96.93	101	96.8427	104	85.64	88.34	89.62	91.74	91.99
i2c	96.92	171	96.8457	175	94.65	96.61	97.17	97.67	97.69
pci_spoci_ctrl	87.84	832	87.9784	823	90.58	93.66	95.47	96.44	96.67
systemcdes	95.00	881	94.9819	885	84.68	91.32	94.81	95.51	95.84
spi	91.12	**1753**	91.1973	**1738**	92.54	93.37	94.04	94.68	95.49
tv80	90.78	4098	90.7896	4092	95.90	96.56	97.47	97.51	97.99
systemcaes	91.92	4207	91.9165	4211	94.99	96.67	98.85	99.09	99.22

to provision for yield optimization by process learning, which may alter or reduce the set of test patterns.

A commercially-viable combinational logic-locking strategy must incur minimal area overhead and negligible impact on the timing paths. The techniques we advocate offer several practical advantages. First, the number of locking sites is a small fraction of a large design — enough to scramble the output response while difficult to disable through mask modifications. Second, the large number of candidate covers shown in Table 2 reveal sufficient flexibility in common netlists to avoid critical timing paths, which we used in our experiments (to preserve critical paths). In fact, heavily-optimized, deep logic designs with inserted control signals may enjoy *many more* candidate covers as the resulting logic signatures will be similar for most test patterns. The MUX insertion strategy is scalable as it only relies on the generation and comparison of logic signatures from initial test vectors (these signatures are not only easy to compute but may be available from DFT tools). Very large designs are typically partitioned to improve testability. Once test vectors are identified for a given partition, identifying MUX candidates depends only on simulating these patterns and comparing logic signatures. Our proposed techniques are also compatible with yield optimization because a sizable variety of test patterns are supported, allowing to add and remove individual patterns as process is optimized for yield. Thanks to numerous candidate locking sites, choosing locks that satisfy a very large number of test vectors up front will allow testing different subsets of vectors without undermining the locking scheme.

6. CONCLUSIONS

Counterfeiting poses serious yet hard-to-quantify risks to the semiconductor industry, whereas successful protection efforts may not lead to easily observable events. Thus, the industry and the research community face significant challenges but also enjoy a range of opportunities. To promote consistent implementation of countermeasures, recent standardization efforts, such as the IEEE 1149.1$^{\text{TM}}$-2013 standard [6], focus on chip authentication and detection of illegally produced chips. But existing standards do not eliminate the exposure of IP to foundries, which can be viewed as a more fundamental challenge than detecting pirated chips.

Effective protection from IC piracy requires integrating several layers of security such as active metering and chip locking, as advocated in [13, 16, 18, 19]. While the use of RSA-based preprocessing in EPIC makes circuit-based key attacks less effective, the original EPIC protocol leaves room for improvement. In this paper, we identify a weakness in combinational circuit locking to attacks based on simulation and/or post-manufacturing test. To eliminate such weaknesses, we propose to restructure EPIC using novel, lightweight IC locking strategy invariant to test response. Compared to traditional combinational XOR-locking, we develop a MUX-locking scheme. Our detailed experiments on IWLS 2005 circuits demonstrate opportunities for such logic locks and confirm the scalability of this procedure to large IC designs. After proposed restructuring of EPIC, chip testing can be performed at the fab before the circuit is unlocked at a trusted facility by the owner of IP rights.[3] This more advanced form of EPIC rules out several classes of attacks and makes contract IC manufacturing more secure. Considering possible attacks against our revised variant of EPIC, we note that a successful attack would also likely work against the previously known versions of EPIC [13, 19], for which attacks have been studied in depth and countermeasures are known [13, 16, 19].

7. REFERENCES

[1] Y. Alkabani, F. Koushanfar, M. Potkonjak, "Remote Activation of ICs for Piracy Prevention and Digital Right Management," *ICCAD* 2007:674-677.

[3] Fabs with different business models (dedicated, pure play, etc) structure testing in different ways. However, yield learning is often an important part of the manufacturing process. To this end, our techniques allow some flexibility in the selection of test vectors. Such flexibility can be increased by inserting additional test points for controllability.

[2] A. Baumgarten, A. Tyagi, J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design & Test of Computers* 2010, 27(1):66-75.

[3] G. Contreras, M. Rahman, M. Tehranipoor, "Secure Split-Test for Preventing IC Piracy by Untrusted Foundry and Assembly," *DFT* 2013, pp. 196-203.

[4] A. Das, et al., "Security Analysis of Industrial Test Compression Schemes," *TCAD* 2013, pp. 1966-1977.

[5] Defense Industrial Base Assessment: Counterfeit Electronics study by U.S. Dept of Commerce Bureau of Industry & Security Office Of Tech. Evaluation, 2010.

[6] H. Horwitz, "IEEE 1149.1$^{\mathrm{TM}}$-2013 Enables Integrated Circuit Counterfeit Protection," *IEEE Today's Engineer* June 2013.

[7] J. Huang, J. Lach, "IC Activation and User Authentication for Security-Sensitive Systems," *HOST*, 2008, pp. 76-80.

[8] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Trans. Info Forensics and Security* 2012, 7(1):51-63.

[9] S. Krishnaswamy, S. M. Plaza, I. L. Markov, J. P. Hayes, "Signature-based SER Analysis and Design of Logic Circuits," *TCAD* 2009, pp. 74-86.

[10] F. Krohm, A. Kuehlmann, A. Mets, "The Use of Random Simulation in Formal Verification," *ICCD* 1996, pp. 371-376.

[11] A. Kuehlmann, "Dynamic Transition Relation Simplification for Bounded Property Checking," *ICCAD* 2004: 50-57.

[12] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," *Tech.Rep.* No. 12-93, EE Dept., Virginia Polytech.

[13] R. Maes et al., "Analysis and Design of Active IC Metering Schemes," *IEEE HOST* 2009:74-81.

[14] A. Mishchenko, R. K. Brayton, J.-H. R. Jiang, S. Jang, "Scalable Don't-care-based Logic Optimization and Resynthesis," *ACM Trans. Rec. Sys. Tech.* (TRETS) 4(4): 34 (2011)

[15] Y. Oren, A.-R. Sadeghi, C. Wachsmann, "On the Effectiveness of the Remanence Decay Side-Channel to Clone Memory-Based PUFs," *CHES* 2013, pp. 107-125

[16] J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, "Security Analysis of Logic Obfuscation," *DAC* 2012:83-89.

[17] J. Da Rolt, G. Di Natale, M.-L. Flottes, B. Rouzeyre, "New Security Threats Against Chips Containing Scan Chain Structure," *HOST* 2011.

[18] M. Rostami, F. Koushanfar, J. Rajendran, R. Karri, "Hardware Security: Threat Models and Metrics," *ICCAD* 2013.

[19] J. A. Roy, F. Koushanfar, I. L. Markov, "Ending Piracy of Integrated Circuits," *IEEE Computer*, October 2010, pp. 30-38.

[20] J. Roy, F. Koushanfar, I. L. Markov, "Protecting Bus-based Hardware IP by Secret Sharing," *DAC* 2008:846-851.

[21] Berkeley Logic Synthesis and Verification Group, "ABC: A System for Sequential Synthesis & Verification," `http://www.eecs.berkeley.edu/~alanmi/abc/`

[22] `http://iwls.org/iwls2005/benchmarks.html`

Appendix: Locking, obfuscation, encryption and scrambling

Recent literature [13, 16, 19] shows a surprising discord of terminology. While [19] and its conference version studied in [13, 16] describe proposed circuit modifications as *combinational locking*, the authors of [16] discuss *obfuscation*, and [8] uses both (following terminology introduced earlier in the FSM context). The analysis in [13] accurately follows original terminology. Yet, the work in [20] *scrambles* busses, and recently these methods have been summarized as *circuit encryption*, since [19] includes RSA cryptography.

Accounting for *idiomatic usage* in the English language, it is surprising to hear that "piracy is best fought with obfuscation," given that *locking* and *encryption* are stronger words that carry concrete meanings and connotations. Representative concepts include *house locks*, *scrambled cable TV signals*, *the obfuscated C contest*, and *Javascript obfuscators*. To find adequate terminology for [8, 13, 16, 19, 20] recall that

locking *preserves the structure, but renders the system temporarily unusable*, until it is *unlocked*

obfuscation *preserves the function but renders the structure unintelligible*, usually with no hope to restore it, hampering reverse engineering and modification

encryption is usually applied to data, rather than executable programs or live circuits, to *render data unusable.* Like *locking*, encryption is reversible by definition, but it preserves neither function nor structure.

scrambling is close to encryption, but easier to undo.

Echoing the calls in [18], we hope that trustworthy terminology consistent with colloquial and technical usage of English words can be established in the field.