IC Piracy Prevention via Design Withholding and Entanglement

Soroush Khaleghi, Kai Da Zhao, and Wenjing Rao
ECE Department, University of Illinois at Chicago, IL 60607, USA
Email: skhale4@uic.edu, zhao68@uic.edu, wenjing@uic.edu

***Abstract*—Globalization of the semiconductor industry has raised serious concerns about trustworthy hardware. Particularly, an untrusted manufacturer can steal the information of a design (Reverse Engineering), and/or produce extra chips illegally (IC Piracy). Among many candidates that address these attacks, Design Withholding techniques work by replacing a part of the design with a reconfigurable block on chip, so that none of the manufactured chips will function properly until they are activated in a trusted facility, where the withheld function is restored back into the reconfigurable block on chip. However, most existing approaches are ad-hoc based, and are facing two major challenges: 1) susceptibility to a category of algorithmic attacks, from attackers in a strong position, such as a manufacturer; and 2) scaling up the defense level is checkmated by the explosion of hardware cost that has to be paid at the designer's side. In this paper, we propose a novel protection scheme, called Entanglement, which can substantially strengthen the Design Withholding framework: 1) the algorithmic attacks are prevented by forcing the attacker to solve a huge number of problems of high computational complexity; 2) the attack cost (in terms of computational complexity) is quantitatively controllable at the designer's end, with low hardware overhead: while the cost of attack can be increased exponentially, the hardware overhead imposed on the designer's side grows only linearly. The proposed work distinguishes itself from the previous works by not relying on the difficulty of finding the solution for some NP-Complete/NP-Hard problems, but rather, on the exponentially boosted number of such problems that an attacker has to solve, while carefully maintaining the growth of the hardware overhead to be scalable via Entanglement.**

I. Introduction

In the past, the IC industry involved the vertical chain of chip manufacturing model, where all the steps, such as design, synthesis, verification, fabrication and test of IC's, were carried out in presumably trustable facilities. However, the continuous decrease in feature sizes imposes the huge cost of upgrading fabrication facilities to meet the growing technological requirements for modern IC fabrication. Furthermore, due to the increased time-to-market pressure for many high-speed and low-power IC's, it is no longer feasible for companies to carry out all the levels of design single-handedly [1]. This has led to the formation of a series of pure contract silicon foundries that specialized in IC fabrication. Consequently, many renowned semiconductor companies have become completely fab-less today.

Globalization of the semiconductor industry has raised serious concerns about trustworthy hardware. Since IC designers no longer have complete control over the manufacturing process, a design is prone to various "hardware attacks", such as *IC Piracy* and *Reverse Engineering* [1], [2]: IC piracy usually refers to an untrusted manufacturer, producing more chips than authorized at a marginal cost, and selling them illegally. Furthemore, an untrusted manufacturer can also steal the design information by employing various reverse engineering techniques.

A strong IC protection scheme must be resilient to a powerful attacker (in the position of a manufacturer), with strong knowledge, tools, and facilities. Similarly to the modern cryptography schemes, hardware security should rely solely on the secrecy of a certain key, rather than the secrecy of the scheme itself. Based on these assumptions, we adopt the following threat models:

- *Who is the attacker? When does the attack happen?* We assume the attacker enters after the creation of the gate-level netlist. It could be any party in the untrusted IC manufacturing chain, which has access to any forms of a design (such as layout, mask, etc.) that is revealed during these stages.
- *What is the goal of the attacker?* We assume that the attacker aims to either gain knowledge of the design (reverse engineering), or produce illegal copies of the functioning IC's (piracy).
- *What are accessible by the attacker? How does it attack?* We assume the attacker to have: 1) the complete knowledge of the gate-level netlist; 2) the power of performing simulation, modifying the design, and manufacturing IC's according to a modified design; 3) full knowledge of the security scheme, except for some "key" that can be kept secret by the designer; 4) access to functional IC's, purchased from the open market, which have been activated by the designer.

The Design Withholding category of techniques work by selecting and replacing a small portion of the design with a reconfigurable block, so that the manufactured chips will not function properly, until they are activated in a trusted facility [8], [10]. Generally, the most powerful way for an attacker to recover the withheld piece is to apply algorithmic attacks on the available part of the design. This actually translates into the practice of solving a number of problems of NP-Complete or higher complexities. We argue that purely relying on the complexity of such problems does not form a strong protection foundation, as these problems might be solvable in a short amount of time under many non-worst-case scenarios.

This work is supported by NSF Grant 1149661.

978-1-4799-7792-5/15/$31.00 ©2015 IEEE

The proposed work in this paper substantially strengthens the framework of Design Withholding by what we refer to as *Entanglement*. The proposed scheme does not rely on the difficulty for an attacker to solve some problems of high complexities, but rather, on the exponentially boosted **number** of such problems that an attacker has to solve. Entanglement gives the designer the full control of scaling up the attacking cost **exponentially**, at a **linearly** increased hardware cost.

Two ways of Entanglement are proposed: 1) the "External Entanglement" technique can exponentially boost the *number* of NP-Complete/NP-Hard problems needed for an attacker to solve, for a *small* withheld function of a design; 2) the "Internal Entanglement" technique decomposes a *large* withheld function into *multiple* pieces, such that the necessitated hardware on the designer's side is efficiently shrunk, while the attacking cost remains huge as that of the original withheld large piece.

II. Previous Works

IC Piracy and Reverse Engineering are highly difficult to address, because of the strong position of an untrusted manufacturer: the manufacturer is in full control of analyzing and modifying the design at the final stage for manufacturing. Unsurprisingly, existing IC protection techniques proposed in the past are mostly passive or ad-hoc solutions.

In the category of *watermarking*-based approaches, a designer's watermark is embedded upon fabrication and cannot be removed from the IC. When an illegal copy of a design is found, the designer will retrieve the watermark in litigation to claim the ownership of that design [3], [4]. Such schemes can passively provide mechanisms for detection of illegal copies, yet cannot *prevent* reverse engineering or IC Piracy from occurring.

Obfuscation-based approaches, on the other hand, aim at "hiding" the design from potential attackers with extra obfuscating hardware, so that no manufactured IC can function correctly, unless being activated by its designer. Since the designer is the only one who knows the correct key, it can control the number of functioning IC's, and prevent untrusted manufacturer from conducting IC piracy [1], [5], [6], [7]. Most obfuscation-based approaches try to ensure that the required effort for an attacker to obtain the correct key is computationally impractical. However, since the entire design, despite being obfuscated, is available to the manufacturer, if the attacker is able to identify the part of the design dedicated for the obfuscation purpose, a functioning IC might be produced by discarding the obfuscating circuitry entirely, thus bypassing the difficult path of searching for the key to unlock the obfuscated design.

The category of *withheld*-based approaches, on the other hand, try to ensure that the entire design is not made available to the manufacture, thus taking away the opportunity for the attacker to gain the full knowledge of the design. Usually, some part of the design is replaced with several lookup-tables (LUT's), which will be configured in a trusted facility after manufacturing of the chips [8], [9]. Another technique in a similar direction works by withholding a part of the wiring topology during the design process, and inserting the correct wiring topology after fabrication [10].

In fact, the obfuscation-based techniques can be covered in the withheld-based framework, making it easier to use the latter to develop theoretical foundation for trustworthy schemes. Furthermore, as opposed to the obfuscation-based approaches where the entire (obfuscated) design is made available the manufacturer, some parts of the design are never given to the manufacturer in the withheld-based approaches. This provides a stronger position for the withheld-based approaches to take away the opportunity for piracy and reverse engineering. Nonetheless, as we will show in the next section, the withheld-based techniques are susceptible to a category of algorithmic attacks, called *ATPG-based* attacks, and are not scalable due to the imposed hardware overhead on the designer's side.

III. Preliminaries and Motivation: Design Withholding Framework

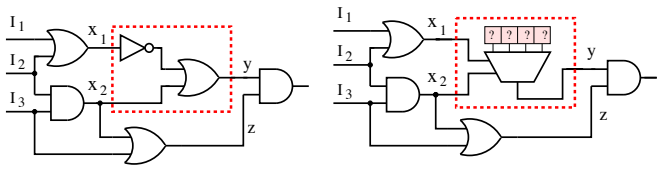
In this section, we provide the models for the Design Withholding framework as a basis to build up the proposed Entanglement schemes. We also present the models and costs on both the attacker's side and the designer's side.

A. Model: Withholding a Single-Output Function

Suppose the circuit in Fig. 1(a) is an original design that needs to be fabricated, with a part of the design (shown in the rectangle) to be withheld from the manufacturer. Since the withheld piece is a Boolean function ($y = \bar{x_1} + x_2$) with 2 inputs and 1 output, it can be replaced by a 4:1 lookup table (LUT) on chip, as is shown in Fig. 1(b). Without the correct content of the LUT, none of the manufactured chips will work as designed, until the LUT is configured inside a trusted facility, according to the withheld function.

To recover the original design, an attacker needs to find the key: the content of the LUT. For the chips on the market that have been activated, there is no direct access for the attacker to probe or observe the securely stored content of the LUT. Nonetheless, the attacker does have access to the primary inputs and outputs of such legally activated chips.

With the help of a fully activated chip and the partially available design (everything except for the withheld part), the attacker can perform an "ATPG-based" attack. For example, in order to find out the value of the first cell in the LUT, the attacker needs to first activate its address by finding an input combination that makes $x_1 = x_2 = 0$. There are two input combinations that can satisfy this condition: ($I_1 = I_2 = I_3 = 0$) or ($I_1 = I_2 = 0, I_3 = 1$). Both patterns can select the value of the first memory cell to the wire y. The next job of the attacker is to make sure that this value at y is propagated to the primary output of the circuit. It turns out that the first



(a) Design with partially withheld piece (b) Chip model

Fig. 1. An example, where a part of a design (including 2 inputs and 1 output) is replaced with a LUT on chip.

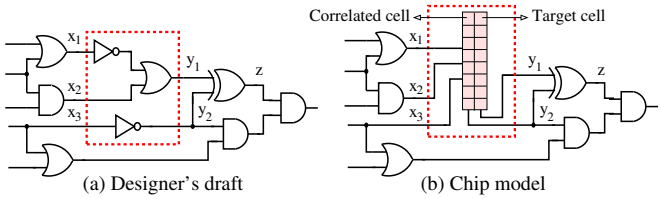


Fig. 2. Withholding a multi-output function to elevate each attack from NP-Complete to NP-Hard complexity

input combination ($I_1 = I_2 = I_3 = 0$) would not work: it will block the propagation of signal y, i.e., the primary output of the circuit would be dominated by the value of the other input bit of the final AND gate (signal z), which is 0. On the other hand, the other input combination ($I_1 = I_2 = 0, I_3 = 1$) can successfully reveal the first bit of the LUT to the primary output.

In general, for every single bit of the LUT, the attacker needs to solve for the primary input combination, such that: 1) the address for the specific cell is selected, and 2) the value of this cell can be propagated (in its original or negated form) to one of the primary outputs. If such an input combination can be found, it can be applied to the primary inputs of the activated chip, and the content of the target cell will be revealed at the output end of the activated chip.

This problem is equivalent to the classical problem of Automatic Test Pattern Generation (ATPG) in IC testing, which is of NP-Complete complexity [11]. In general, for every single bit of the LUT, an attacker has to perform such an "ATPG-based" attack, with the goal of finding a certain combination of the primary inputs to "stimulate" a cell, while at the same time, "propagate" the cell's content to one of the primary outputs. This process can be done for every cell in parallel, to finally recover the withheld function of the design.

B. Model: Withholding a Multiple-Output Function

Fig. 2(a) shows an example of withholding a multiple-output function with 3 inputs (x_1, x_2, x_3) and 2 outputs (y_1 and y_2). Accordingly, a LUT with 16 memory cells is required to replace the 2-output withheld function (8 cells for each output), as is shown in Fig. 2(b).

We argue that the change from a single-output to a multiple-output function has made a qualitatively different problem to solve for an attacker. This is due to the *correlation* between the multiple output bits (y_1 and y_2). For example, as is shown in Fig. 2(b), in order to find the value of the right column of the first address in the LUT (shown as the "Target cell"), the attacker needs to solve the primary inputs to: 1) activate the address of $x_1 = x_2 = x_3 = 0$, and 2) propagate the value of the Target cell from y_1 to the primary output of the circuit. However, any input combination that activates the Target cell at y_1 would also activate the cell of the left column at y_2 (shown as the "Correlated cell") at the same time. The value of this cell is unknown to the attacker, despite the fully specified primary inputs. Due to the correlation between these two cells, an attacker cannot solve each of them independently, or in parallel. Instead, each has to be modeled as a distinct unknown variable in the ATPG algorithm to be solved at the same time.

Propagating the value of the Target cell in the presence of the unknown values of many Correlated cells is qualitatively different, and a harder problem to address, because keeping track of all the unknown values' symbolic computation simultaneously will quickly become intrackable as the number of unknown values increases. Alternatively, if the attacker does not keep each unknown as a dedicated variable, the computation will quickly lose precision, because to too many signals of unknown values are mingled together. For example, the attacker cannot determine the output of the XOR gate (signal z), because both of the inputs of this gate (y_1 and y_2) have unknown values. In other words, as opposed to the case of a single-output function, the attacker cannot propagate the value of the Target cell (y_1) to the next level (z), due to the unknown Correlated cell (y_2) that is not accessible by the attacker.

Such an ATPG problem with unknown values is of NP-Hard complexity [12], and is significantly harder than the single-output function case, which is of NP-Complete complexity. Furthermore, it is also shown that most existing deterministic ATPG tools are not able to handle such tasks efficiently [12].

C. Challenges for Design Withholding Framework

In this section, we provide the cost analysis for the designer (in terms of hardware) and for the attacker (in terms of computational complexity) to crack the Design Withholding scheme.

Assuming that the withheld function has n inputs $\{x_1, x_2, ..., x_n\}$ and m outputs $\{y_1, y_2, ..., y_m\}$, it can be modeled by an LUT with n selection lines (addressing to 2^n memory cells), and m output lines. Accordingly, $2^n \times m$ memory bits are needed, in addition to the MUXes, constituting the hardware cost for the designer. On the other hand, an attacker has to solve $2^n \times m$ problems, each with NP-Complete (for $m = 1$) or NP-Hard (for $m > 1$) complexity.

However, it is dangerous for a protection scheme to rely solely on the hardness of NP-Complete/NP-Hard problems, per se, because even though such problems can take exponentially scaled time to solve in the worst case, they could be easily solvable in some of the best cases, when constraints are not stringent [13]. Consequently, there is no guarantee that an attacker, aided by powerful ATPG tools, cannot obtain the desired information within a reasonable time limit.

In order to achieve a theoretically sound barrier, the designer should rely on the *number* of NP-Complete/NP-Hard problems for an attacker to solve, rather than the difficulty of solving the problems itself. Scaling up the number of such problems essentially means to increase the number of memory cells to crack. Since the memory stores the truth table of the withheld function, the increase in the number of memory cells is exponential to the size of the withheld function. However, the hardware cost for the designer to implement the withheld piece grows at the same exponential rate, making it unrealistic for a designer to bear the cost.

As an example, if the designer wants to double the number of ATPG-based attacks by withholding one more input signal (thus doubling the truth table of the original plan), the size

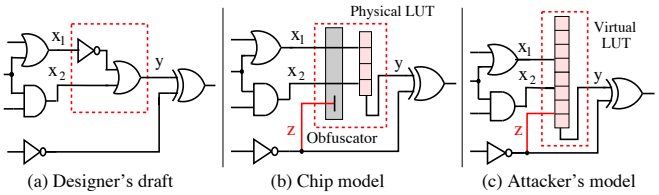


Fig. 3. External Entanglement: a) the original withheld function; b) designer's cost (LUT size) kept low with the help of an Obfuscator; c) the attacker has to solve a much larger number of cells, due to the entanglement

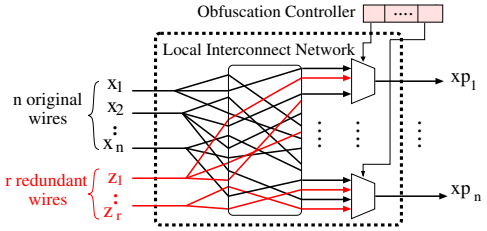


Fig. 4. General structure of a programmable Obfuscator

of LUT would double to be $2^{n+1} \times m$. Such a doubling in search space of the attacker is achieved at the cost of doubling the hardware on each chip. This is apparently not a scalable approach to deliver a desired level of security.

IV. Entanglement

In this section, we propose two *Entanglement* techniques: 1) drastically increase the cost of the attacker for a **small** withheld function, without boosting the hardware overhead; 2) drastically decrease the hardware cost of a **large** withheld function, while maintaining the high cost to attack. In both schemes, the computational complexity for an attacker to recover the withheld information is quantitatively controllable at the designer's end. Furthermore, while the cost of attack can be scaled up exponentially, the hardware overhead grows only linearly on the designer's side.

A. External Entanglement

As we discussed in the previous section, an ideal protection scheme must force an attacker to recover a huge truth table, while the imposed hardware overhead on the designer should be much less. The main idea of achieving such a goal is by introducing some "noise" or redundancy, such that one can virtually enlarge the search space for an attacker. This can be achieved if the attacker cannot distinguish between the added redundant part ("noise") and the original withheld piece ("signal"). In other words, if the attacker lacks some crucial information to identify the small subset of "signal" among the "noise", it will have to treat them the same way and solve them all. Meanwhile, the designer, with the full knowledge of the signal/noise distinction, is able to pay the small cost with respect to the signal part only. If such a scheme can be developed at a low hardware overhead, it can successfully achieve the goals of IC piracy and reverse engineering prevention.

Fig. 3 shows an example, for which, the withheld piece is a function with 2 inputs (x_1 and x_2) and 1 output (y). As is shown in Fig. 3(b), the 2 original inputs (x_1 and x_2) and a redundant "noise" signal z are all fed into a programmable *Obfuscator* logic block. The role of the Obfuscator is to block the noise, signal z, while propagating the original signals (x_1 and x_2) for an activated chip. By withholding the configuration of Obfuscator from the attacker, it will have to work on a virtually enlarged function of 3 inputs, thanks to the additional noise signal z, which "entangles" the original function of x_1 and x_2. The Obfuscator block should also be programmed in a trusted facility, so that the original address bits (x_1 and x_2) can be "disentangled" for activating the chips. In this scheme, the size of the reconfigurable block will remain unchanged (a single-output LUT with 4 cells in this example).

Since the attacker does not have on-chip access to the Obfuscator block, it cannot identify which of the 3 wires (among x_1, x_2, and z) is the redundant one. Therefore, the attack model has to model the much larger virtual function of 3-bit input. As is depicted in Fig. 3(c), the total search space is enlarged to be 8 bits in this case, effectively doubling the attacker's cost without doubling the necessitated LUT size on the designer's side.

Fig. 4 shows a general implementation of the Obfuscator block, where n original wires and r redundant ones are entangled. This block can be implemented with n MUXes, selecting from a few inputs either form the n original address bits or from the r redundant signals to output to an address bit of the LUT. As long as a permutation of original wires (x_1 to x_n) can be obtained at the outputs of these n MUXes, the network can be implemented with local connections. After manufacturing, the value of the *Obfuscation Controller* will be set (together with the LUT) in a trusted facility, in a way such that: 1) all the redundant noise signals are blocked; and 2) a permutation of the original wires are selected for the LUT. Now, the "key" for the design consists of two parts that must be stored in a protected memory on chip: the content of the LUT, and the content of the Obfuscation Controller.

Cost Analysis: The hardware cost in the general case of a withheld function with n inputs, m outputs, and r redundant noise signals includes: the implementation cost of the LUT, of $O(m \times 2^n)$ complexity, plus the cost for the Obfuscator block, of $O(n + r)$ complexity (including n MUXes and the interconnect network). Since an attacker cannot identify which of the $n + r$ wires are the original ones, the attack model has to tackle an enlarged virtual LUT of $n + r$ address bits. This effectively boost the total search space to be $O(m \times 2^{(n+r)})$, which is 2^r times larger than the complexity of the hardware cost imposed on the designer's side. Therefore, such an External Entanglement scheme achieves the goal of blowing up the attacking cost exponentially, while maintaining the hardware cost on the designer's side to grow linearly only.

B. Internal Entanglement

Fig. 5(a) shows a single piece of design with 6 inputs (x_1 to x_6) and 1 output (y). The total number of memory cells required to implement this function is $2^6 = 64$. This figure also illustrates how the target function can be divided into multiple parts, while all of them are kept entangled so that the cost of attacking cannot be reduced. The original function is divided into two layers: the outputs of the two pieces in the first layer (y_1 and y_2) are fed to the one piece in the second

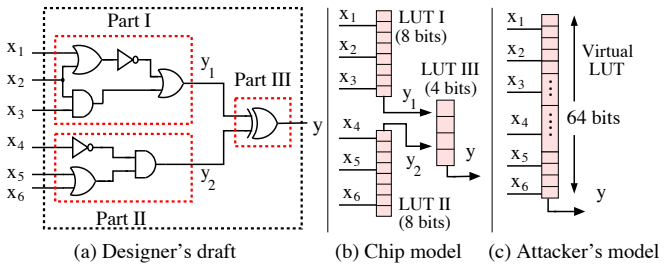


Fig. 5. Internal Entanglement: a) a large withheld piece is decomposed into 3 smaller pieces; b) Using three interlocking small LUT's to shrink the hardware cost; c) the large virtual LUT remained for attacker to solve

layer. In this example, the withheld pieces in the first layer are two functions, each with 3 inputs and 1 output. The withheld piece in the second layer is a function (XOR gate) with 2 inputs and 1 output. Fig. 5(b) shows the on-chip implementation of the withheld pieces using three small LUT's. The hardware cost, in terms of the total number of cells in all the LUT's, is reduced to 20.

The interlocking way of connecting these LUT's in 2 levels essentially forms an entanglement such that the attacker has no way of activating and observing the value of a particular cell in any of the LUT's. Basically, the cells of the two LUT's in the first layer (y_1 and y_2) serve as the address bits of the LUT in the second layer. Accordingly, the attacker is only able to select an address at the first layer to activate the outputs at y_1 and y_2. At this point, the attacker loses control to activate any selected cell in the second layer, due to the fact that the values of the cells in the first layer (y_1 and y_2) are needed to proceed, yet they remain unknown. Furthermore, assuming that the attacker selects one cell in each of the LUT's in the first layer by sending the required values to signals x_1 to x_6, and observes the value at signal y for the activated chip, it does not reveal the content of any of the cells in any of the LUT's. This is due to the fact that the values of the cells in the first layer (y_1 and y_2) are unknown, even though the cells containing those values are selected by the attacker. Accordingly, the observed value at y could belong to any of the cells in the second layer. Therefore, as is depicted in Fig. 5(c), the attacker has to model the entire system as one big LUT with 6 inputs (x_1 to x_6), and $2^6 = 64$ cells to solve, while the hardware cost is shrunk to be 20.

In general, the Internal Entanglement scheme employs two or more layers. In the case of having two layers, there are k LUT's at the first layer, with possibly different sizes (number of address bits), where the outputs of the LUT's at the first layer are connected to the address bits of a single LUT at the second layer. This can be easily extended to more than two layers of LUT's. Generally, each LUT could be a single-output or a multiple-output function.

Cost Analysis: Suppose there are k one-output LUT's at the first layer, each with n address bits, and the outputs of these LUT's are the address bits of a one-output LUT in the second layer. While the hardware cost on the designer's side is $O(k \times 2^n + 2^k)$, the key size that an attacker has to crack is $O(2^{n \times k})$. Therefore, the Internal Entanglement scheme drastically reduces the hardware cost at the designer's side, without affecting the attacking cost.

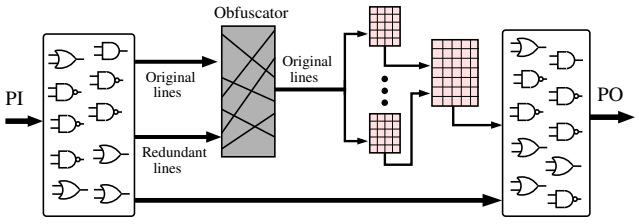


Fig. 6. Applying both External Entanglement and Internal Entanglement

Overall, although each of the two proposed techniques can be implemented independently by itself, they can be combined to form an even stronger overall protection scheme. Fig. 6 shows a schematic design, where both entanglement techniques are combined together.

V. Simulation Results

The effectiveness of the proposed scheme is analyzed using the ISCAS-85 combinational benchmarks. Attacks are simulated using the Atalanta ATPG tool [14].

Fig. 7 verifies the fact that a strong protection scheme should not rely on the hardness of some NP-Complete problems, by showing the runtime of various cases for the attacker. The attacks are performed for each benchmark, on all the possible single-output withheld functions. As the figure indicates, even though it could take a relatively long time in some of the worst cases, most of the cells can be cracked in a very short time. While some of the worst cases could take up to 250ms to solve, the median runtime of all the benchmarks is as little as 27.7ms. Apparently, even though NP-Complete problems can take exponential time to solve in the worst case, the average cases are very easy to solve in the cases of ATPG-based attack.

Fig. 8 verifies the effectiveness of the Internal Entanglement scheme. It shows that as the hardware overhead on the designer's side grows linearly, the runtime (measured by the number of cells to solve) for an attacker increases exponentially. To verify a reasonable performance, we examine various hardware overhead, ranging from 2.5%, to 25% of the total number of transistors. The values of k and n for the Internal Entanglement scheme are selected to maximize the key size

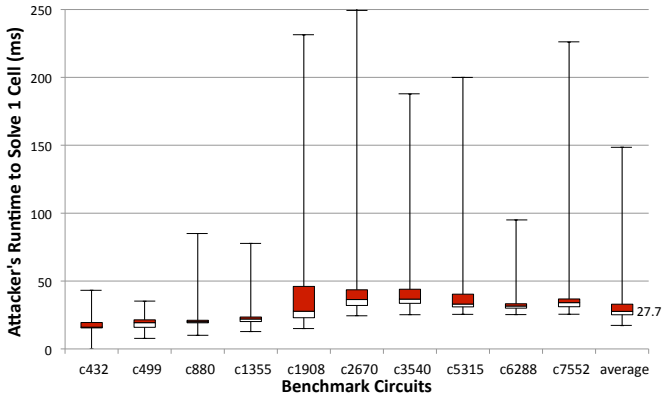


Fig. 7. The minimum, first quartile, median, third quartile, and maximum runtime for an attacker to solve an input pattern for each cell, in the case of withholding one piece of design with a single-output (NP-Complete complexity) over all possible single-output functions

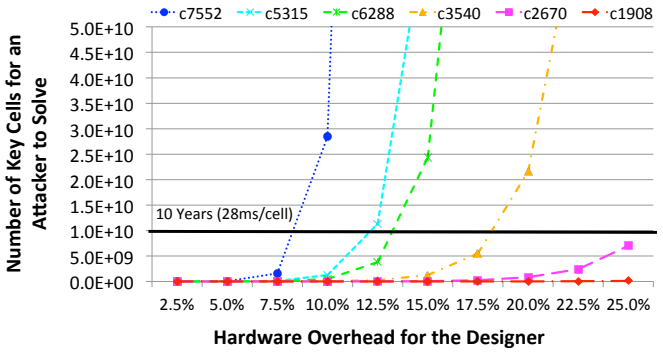


Fig. 8. Attacking cost vs. Hardware cost for the Internal Entanglement scheme

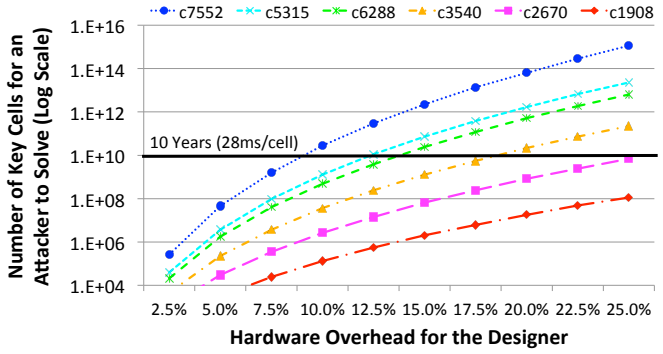


Fig. 9. Attacking cost **(Logarithmic Scale)** vs. hardware overhead for the Internal Entanglement scheme

for the attacker. The time scale (10 years line) is calculated using the median time obtained in Fig. 7 (28ms per cell). Even if an attacker can employ much faster computers and perform the cracking process in parallel, the attack complexity grows exponentially with a linearly increased hardware overhead, as is verified by Fig. 9, which shows the same data in a logarithmic scale.

Fig. 9 shows clearly that, as the size of the circuit becomes larger, the required hardware cost to achieve computationally impractical attacks becomes smaller. For example, while 20% hardware overhead is required to achieve 10 years of attacking time for C2670, a 10% overhead for C7552 is sufficient to achieve a much better protection (more than 1000 years of computation). Overall, as long as the attacker has to spend a reasonable amount of time to solve each cell (which is a valid assumption for the NP-Complete problems), the protection level is fully controllable by the designer, under a very reasonable amount of hardware overhead.

VI. Conclusions

Two Entanglement schemes are proposed in this paper, for the withheld-based framework: 1) the External Entanglement scheme forces the attacker to solve a hugely boosted number of problems for a small withheld piece, at a low hardware overhead for the designer; and 2) the Internal Entanglement scheme decomposes a large withheld function into multiple ones, such that the hardware overhead is drastically reduced for the designer, while the cost to attack remains that of the original large withheld function. The proposed techniques in this paper aim at defending the design against the very powerful and effective ATPG-based attacks, thus pushing an attacker to resort to much harder strategies such as side-channel attacks [4], [15] [16]. We show that by engaging Entanglement in the withheld-based framework, the ATPG-based attacks can be made arbitrarily expensive with the designer's full control. Meanwhile, the Entanglement guarantees that the exponentially scaled up attacking cost is feasible to achieve: the needed hardware cost at the designer's end only increases linearly. This scheme has laid a solid foundation for withheld-based protection schemes against ATPG-based attacks, and provided a game-shifting paradigm to strengthen the weakest defense against IC piracy and reverse engineering attacks.

References

[1] J. Roy and F. Koushanfar, "EPIC: Ending Piracy of Integrated Circuits", *Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[2] R. Torrance and D. James, "The State-of-the-art Semiconductor Reverse Engineering", *IEEE/ACM Design Automation Conference*, pp. 333–338, 2011.

[3] A. Kahng, J. Lach, W. M. Smith, S. Mantik, I. Makrov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe, "Watermarking Techniques for Intellectual Property Protection", *IEEE/ACM Design Automation Conference*, pp. 776–781, 1998.

[4] M. Rostami, F. Koushanfar, J. Rajendran and R. Karri, "Hardware Security: Threat Models and Metrics", *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 819–823, 2013.

[5] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri, "Security Analysis of Logic Obfuscation", *IEEE/ACM Design Automation Conference*, pp. 83–89, 2012.

[6] Y. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security", *USENIX Security*, pp. 291–306, 2007.

[7] R. S. Chakraborty and S. Bhunia, "Hardware Protection and Authentication through Netlist Level Obfuscation", *IEEE/ACM Conference on Computer-Aided Design*, pp. 674–677, 2008.

[8] A. Baumgarten, A. Tyagi and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers", *IEEE Design and Test Computers*, vol. 27, pp. 66–75, 2010.

[9] L. Bao and B. Wang, "Embedded Reconfigurable Logic for ASIC Design Obfuscation against Supply Chain Attacks", *Design, Automation and Test in Europe*, pp. 1–6, 2014.

[10] S. Zamanzadeh and A. Jahanian, "Automatic Netlist Scrambling Methodology in ASIC Design Flow to Hinder the Reverse Engineering", *IFIP/IEEE Very Large Scale Integration*, pp. 52–54, 2013.

[11] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[12] D. Erb, M.A. Kochte, M. Sauer, H.Wunderlich and B. Becker, "Accurate Multi-cycle ATPG in Presence of X-Values", *Asian Test Symposium (ATS)*, pp. 250–245, 2013.

[13] H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems for Combinational Logic Circuits", *IEEE Transactions on Computers*, pp. 555–560, 1982.

[14] H. Lee and D. Ha, "An Efficient Forward Fault Simulation Algorithm based on the Parallel Pattern Single Fault Propagation", *Proc. of IEEE International Test Conference*, pp. 946–955, 1991.

[15] M. Stanojlovic and P. Petkovic, "Strategies Against Side-Channel Attack", *Small Systems Simulation Symp*, pp. 86–89, 2010.

[16] M. Joye, "Basics of Side-channel Analysis", *Cryptographic Engineering*, pp. 365–380, 2009.