Hardware Obfuscation Using Physically Unclonable Functions Against IC Piracy

BY

SOROUSH KHALEGHI
B.S., Sharif University of Technology, 2012
M.S., University of Illinois at Chicago, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

To Sepideh,

And my parents, Sima and Hassan.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Wenjing Rao for everything she has done for me. Her encouragement, inspiration, patience, and valuable hints and advice helped shape this work. Her considerations for both keeping in mind the big picture of research and digging deep into the technical details, as well as her emphasis on effective writing and presentation were critical in every step of my Ph.D. I always enjoyed the regular research meetings with Dr. Rao on various research ideas, as well as our conversations on a broader range of topics. Over the past six years, she has taught me a great deal, both professionally and personally. Particularly, her contagious positive attitude, which in my opinion was the special ingredient in making her a great advisor, is a lesson that I will never forget.

I want to extend my appreciation to Dr. Zhichun Zhu, Dr. Zhao Zhang, Dr. Igor Paprotny, and Dr. John Lillis for serving on my dissertation committee. I would also like to thank Dr. Amit Ranjan Trivedi for his guidance during my preliminary examination.

I also want to thank all my wonderful professors and teachers over the many years of my studies. Particularly, I want to thank Dr. Yue Yin for her great support and guidance on our educational collaboration focusing on computational thinking. I would also like to thank Dr. Natasha Devroye for her valuable insights that helped me get a deeper understanding of my research topics by looking at them from a different perspective.

I shared many pleasant moments with my fellow group members over the years: Soumya Banerjee, Jian Xu, Paolo Vinella, and Kai Da Zhao. Their feedback and insights helped improve this work.

ACKNOWLEDGMENTS (Continued)

I always consider myself to be very lucky for having many wonderful friends. I want to thank them all for creating the most meaningful moments of my life. Particularly, I want to thank Alborz Rezazadeh, Ali Javadi, Amir Hosseinipour, Bahareh Abbasi, Dorsa Fahimi, Ehsan Asgari, Hadi Pouransari, Hossein Mirghasemi, Mehdi Sharifzadeh, Mohammad Bavarian, Nasim Mobasheri, Sadegh Salehi, Zia Ghiasi, and Zaeim Mehraban.

I am forever indebted to my parents for their devotion and everlasting love. My mother, Sima, has devoted her entire life to raising me and my three siblings to be happy and successful. No words can describe how thankful I am to her. I often wish that I had made the movie “It’s a wonderful life”, so that I could dedicate it to her, for all her sacrifice. I am also grateful to my father, Hassan, who taught us to always seek and do the right thing by his own way of life. I owe a lot of who I am to him, for his wisdom and support. I would also like to thank my siblings. My older brothers, Sina and Salman, have made every single step of my life a lot easier than it was for theirs. They have been sources of comfort from far away by offering me their continuous help, support, guidance, and inspiration. I am also thankful to my dear little sister, Dr. Saba, for her love and support and for being on my side my entire life. My thanks also goes to my sister-in-law, Sanam, for her support and generosity. Sanam and Sina opened their warm home to us for an entire summer while expecting my lovely nephew, Kian.

Finally, I want to thank the love of my life, the kindest person I have ever known, and my best friend: Sepideh. None of this would have been possible without your endless love, support, and patience. Thank you!

SK

PREFACE

This dissertation is an original intellectual product of the author, Soroush Khaleghi. All of the work presented in this dissertation was performed at the University of Illinois at Chicago. The work has been presented and published in (Khaleghi et al., 2015), (Khaleghi et al., 2016), (Khaleghi and Rao, 2018). The research work was funded by NSF Grant CNS-1149661

Soroush Khaleghi
October 22, 2018

CONTRIBUTION OF AUTHORS

The contents of Chapter 2 was published in (Khaleghi et al., 2015). Wenjing Rao, my adviser, was the lead investigator in this project. I was responsible for coming up with the idea, formalizing the problem, proposing the solutions, designing the experiments, and writing of the paper. Kai Da Zhao was responsible for performing the experiments and analyzing the data.

The contents of Chapter 3 was published in (Khaleghi et al., 2016). Wenjing Rao, my adviser, was the lead investigator in this project. I was responsible for conceptualizing the idea, formalizing the problem, designing the framework and writing of the paper. Paolo Vinella was responsible for performing the experiments as part of his M.Sc. thesis at University of Illinois at Chicago. Soumya Banerjee contributed to manuscript edits.

The contents of Chapter 4 was published in (Khaleghi and Rao, 2018). Wenjing Rao, my adviser, was the lead investigator in this project. I was responsible for coming up with the idea, formalizing the problem, proposing the solutions, designing and performing the experiments, as well as writing of the paper.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

FIGURE **PAGE**

LIST OF ABBREVIATIONS

IC	Integrated Circuit
IP	Intellectual Property
NP	Nondeterministic Polynomial
LUT	Look-up Table
ATPG	Automatic Test Pattern Generation
CRP	Challenge-Response Pair
PUF	Physically Unclonable Functions
ECC	Error Correcting Code
MUX	Multiplexer
IEEE	Institute of Electrical and Electronics Engineers
UIC	University of Illinois at Chicago

SUMMARY

The goal of this thesis is to establish a theoretical foundation and defensive mechanisms against integrated circuit (IC) piracy. IC piracy is defined as the practice of an untrusted manufacturer to produce illegal copies of IC chips, or to steal the intellectual property of the IC design via reverse engineering approaches. IC piracy prevention is especially challenging, as the potential attackers are in the very strong position of chip manufacturers, having accesses to the design details and controls to the final production process.

The proposed work aims at the *hardware obfuscation* based prevention strategy: an “obfuscated” IC design given to the untrusted manufacturer will yield chips that are “locked” (non-functioning), until being “unlocked” (configured correctly) in a trusted facility. The obfuscation strategy mimics that of an encryption process, ensuring that some critical information (analogous to the key) of the design is not revealed to the untrusted manufacturer. During the “unlocking” process (analogous to decryption) in a trusted facility, the key is used to restore the chips to their correct functionality.

Currently, many obfuscation approaches exist, yet they mostly present various ad-hoc choices of obfuscation target, and are based on heuristic methods. There lacks a theoretically sound and provably secure foundation to address the two main categories of attacks: 1) *algorithmic attacks* applied on the obfuscated design, that could potentially crack the keys efficiently, and 2) *physical attacks* applied on the unlocked chips, aiming at reading out the keys directly from the on-chip memory cells.

The goal of this work is to achieve an IC piracy prevention paradigm similar to modern cryptography, in their reliance on the secrecy of a key alone, rather than that of the scheme itself, as well as their provable defense strength via imposing prohibitively high attacking cost, measured by computational complexity. This will lay the basic principles for a future “Design-Against-Piracy” paradigm, similar to the widely used “Design-For-Test” practices in IC industry today.

The proposed work aims at simultaneously *expanding the control of a designer*, and *restraining the control of an attacker*, throughout the IC design and fabrication process. This is approached in the following ways: 1) novel primitives are introduced with the representational power to unify the existing obfuscation schemes, and based on which quantitative analysis of attack and defense costs are possible to carry out; 2) by systematic entangling the various obfuscation primitives in a hierarchical manner, a strong defense mechanism can be built to ensure that the cost of algorithmic attacks (in terms of computational complexity) can be raised exponentially, while the designer’s cost (in terms of hardware overhead on chip) only increases linearly; 3) against the worst-case scenario of combined algorithmic and physical attacks, a preventive architecture is proposed to deliver a unique key per chip, via the engagement of Physically Unclonable Functions (PUFs) into the obfuscation paradigm. This will ensure that even a completely leaked key cannot be used for piracy purposes. Overall, the deliverable security (in terms of attack costs) and defense costs (as overhead on the designer’s side) can be quantitatively modeled, analyzed, and proved, in an asymptotic manner, making it suitable for the scalability of IC design, serving a strong basis for a “design against piracy” framework.

The research work in this thesis will deliver an overall strong foundation of hardware security to actively prevent IC piracy with the following guarantees:

1. Any attackers (even in the strongest position of a manufacturer) cannot crack the design or unlock the chips within a reasonable amount of time, and such attacking cost is in full control of the designer.
2. Any wrong key cannot unlock a chip to function, and even in the extreme cases of a completely leaked key, the security of the original design can be nonetheless protected, as the unique key one chip can neither unlock any other chip, nor be used to reverse engineering to gain information of the original design.

CHAPTER 1

INTRODUCTION

Parts of this chapter have been presented in (Khaleghi et al., 2015), (Khaleghi et al., 2016), (Khaleghi and Rao, 2018). Copyright © 2015, 2016, 2018, IEEE.

1.1 Overview

The goal of the proposed work is to develop a theoretically sound foundation against the threat of integrated circuits (IC) piracy (Rostami et al., 2013) (Roy and Koushanfar, 2008), (Torrance and James, 2011). IC piracy usually refers to an untrusted manufacturer producing more chips than authorized to sell at a marginal cost, or stealing the intellectual property of the design via reverse engineering techniques. These threats have emerged due to the globalization of the semiconductor industry. According to the Semiconductor Industry Association (SIA), counterfeiting costs U.S semiconductor industry over $7.5 billion per year (Office, 2011).

Using the paradigm of modern cryptography as an analogy, if the original IC design file is considered as a plain-text "message", then a successful prevention framework should ensure that security relies solely on the secrecy of a key, not that of the scheme or the encrypting algorithms. Similarly, the strength of a defense mechanism should rely on imposing a prohibitive attacking cost: cracking of a key is provably infeasible in a limited time.

However, the reality of IC piracy prevention is far from achieving such a goal. Most existing approaches are ad-hoc based, lacking of theoretical principles, fine-grained quantitative models, and provable results. Furthermore, there lacks a similar foundation which serves as the basis to the approaches, as in the case of modern cryptography paradigm. This is partially because the challenges of IC piracy are relatively new, but more importantly, IC piracy has its unique challenges that have no parallels in the software domain.

Essentially, IC piracy has to deal with a very strong potential attacker in the position of an untrusted manufacturer, who is usually at the final stage of the production chain to fabricate the chips, and has access to the majority, if not the entire design in details [1].

Furthermore, an IC piracy prevention framework cannot be achieved by a straightforward adoption of the established encryption schemes. **First**, one cannot simply apply, say, RSA (Rivest et al., 1978) encryption, onto a design file (treating it as a plaintext message), because then the encrypted file becomes useless for fabrication of any chips: the transformation imposed by RSA (and other encryption algorithms) onto the original design file does not yield a valid format of description of any chip, let alone a desired "locked chip". In the context of IC piracy prevention, since the design files are used to describe a chip product (which has a well-defined functionality), and the potential attacker is in the position of the manufacturer, a "properly encrypted" design file should still yield the valid, intended chip products, except that these chips (manufactured according to the "encrypted" design file) should be "locked" in their functionalities, until some key is applied to "unlock" them. Such a requirement (encryption of a design yielding locked chips) is the unique attribute of the IC piracy prevention paradigm. **Second**, the abilities and tools of the attacker in the position of manufacturer are significant: the potential attackers

[1]It is true that the format of the design that is made available to the manufacturer (such as the final layout form) might be quite different from a human-understandable one (such as the high-level behavioral description), yet powerful tools are available to perform reverse engineering practices to extract design information and steal intellectual properties (Torrance and James, 2011) (Chipworks, 2012), (DARPA, 2012), (Fleet and Dransfield, 1998).

are essentially endowed with the job of fabricating the chips, so they should always have access to most (if not all) details of the design, and controls to the process. In addition, they have possess powerful tools to perform side channel attacks, or invasive attacks, physically, on legally unlocked chips. **Finally**, if a *common* key is used across all the fabricated chips, then such a key (that must be stored on every chip) becomes the most vulnerable part of the entire obfuscation mechanism. In other words, a leaked key from side-channel attack can be used to unlock other chips, thus compromising the entire security mechanism.

1.2 Previous Works

Against IC piracy, a number of approaches have been proposed to embed in the design or on chip some forms of a designer's watermark, or a buyer's signature, such that illegal copies of a design or unauthorized chips can be detected and used in litigation to prove the ownership of that design or the source of piracy (Kahng et al., 1998a), (Koushanfar et al., 2005), (Kahng et al., 1998b), (Lach et al., 1998), (Rostami et al., 2013), (Caldwell et al., 2004), (Holcomb et al., 2009), (Ruhrmair et al., 1011), (Rostami et al., 2013). Such schemes can *passively detect* IC piracy practices, yet cannot *actively prevent* IC piracy from occurring. Although watermarking and fingerprinting schemes provide mechanisms for detection of illegal copies in case of litigation, they cannot prevent reverse engineering or unauthorized manufacturing of a design in the first place.

Obfuscation-based approaches (Roy and Koushanfar, 2008) (Rajendran et al., 2012b) (Alkabani and Koushanfar, 2007) (Chakraborty and Bhunia, 2008) (Chakraborty and Bhunia, 2009) (Rostami et al., 2013) (Baumgarten et al., 2010) (Khaleghi et al., 2015) (Zamanzadeh and Jahanian, 2013) follow such a preventive flow, with the main idea of "concealing / withholding" an important part of the design

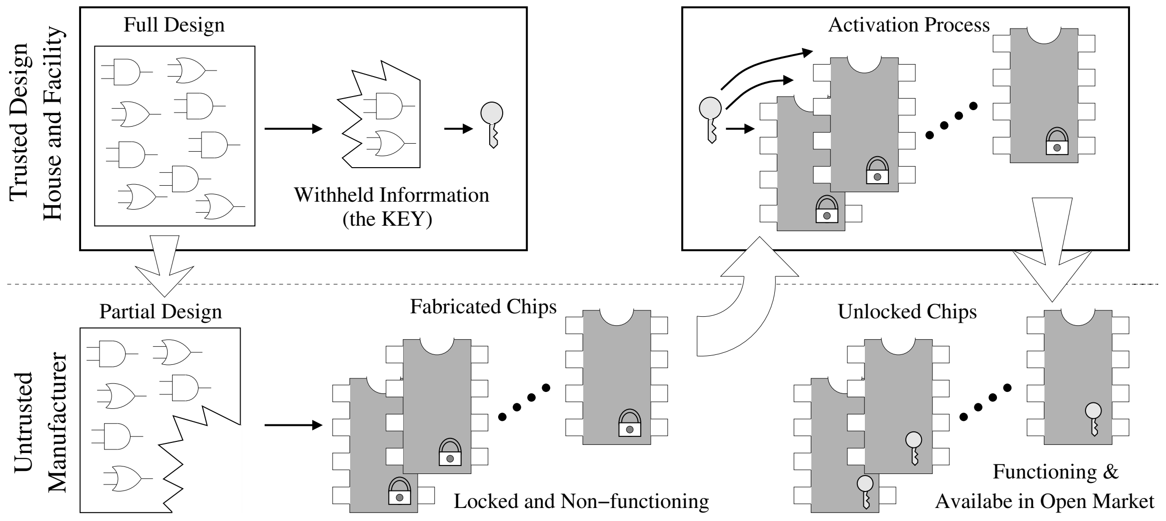
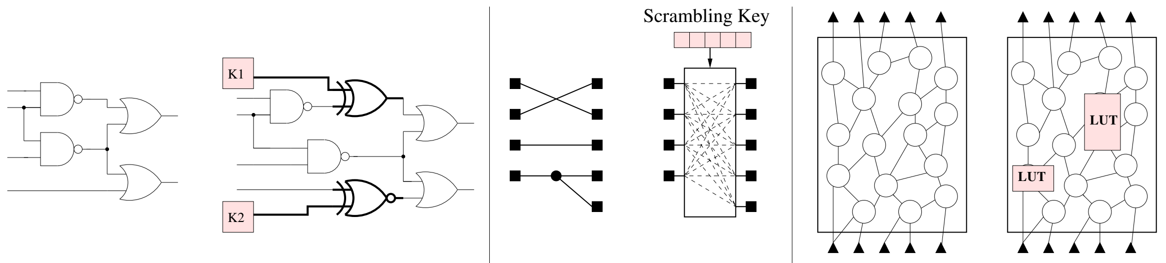

Trusted Design House and Facility
Full Design
Withheld Inforrmation (the KEY)
Activation Process
Untrusted Manufacturer
Partial Design
Fabricated Chips
Locked and Non−functioning
Unlocked Chips
Functioning & Availabe in Open Market

Figure 1: Illustration of the design-manufacturing flow assumed by the obfuscation-based schemes

(which essentially constitutes the "key"), so that no manufactured chip can function correctly (thus considered "locked"), until being "activated" in a trusted facility, by restoring the missing part of the design into the chips. Then, the chips are considered "unlocked" and can be made available to the open market. As long as no direct access is available to read out the re-installed keys in the unlocked chips, an attacker cannot restore the entire design, or deliver any functioning chips without the knowledge of the key. Since the designer is the only one with the withheld information (thus the entire design) and controls the unlocking process for every chip, untrusted manufacturers are prevented from conducting IC piracy. Fig. 1 illustrates the overall flow of such obfuscation based schemes.

Fig. 2(a) shows an example of an obfuscation scheme from (Roy and Koushanfar, 2008)(Rajendran et al., 2012b), which allows the designer to configure the negation of any wires in a post-manufacturing stage. This is achieved by inserting additional XOR/XNOR gates, called *key gates*, on selected lines



(a) key gates (b) Wire scrambler (c) LUT insertion

Figure 2: (a) The content of K1 and K2 obfuscates the functionality of the circuit; (b) wiring interconnects are scrambled for obfuscation purposes; (c) obfuscation by replacing logic functions with LUTs.

of the original design, and use a 1-bit memory cell to control the negation of the other input of the key gate. The key gate thus acts as a configurable NOT gate, depending on the bit stored in the memory. For a manufactured circuit to function as the original design, the content of the memory cells connected to the key-gates must be configured correctly in the post-manufacturing stage ($K1 = 0$ and $K2 = 1$ in the case of Fig. 2(a)). Such memory configuration becomes the key and should be kept as a secret from the potential attackers (untrusted manufacturer).

Fig. 2(b) shows a different example of obfuscation, named *scrambling*, where the designer can choose to withhold key information about any interconnect configurations (Zamanzadeh and Jahanian, 2013). Fig. 2(b) shows the original connections of some wires, and then obfuscation is performed by replacing the interconnect configuration with a *Scrambler*. The scrambler in this example allows the 4 input lines and the 5 outputs to be connected in arbitrary ways, so that the correct configuration remains unknown to the attacker without having access to the content of the scrambling key, which will be applied after the manufacturing of the chips.

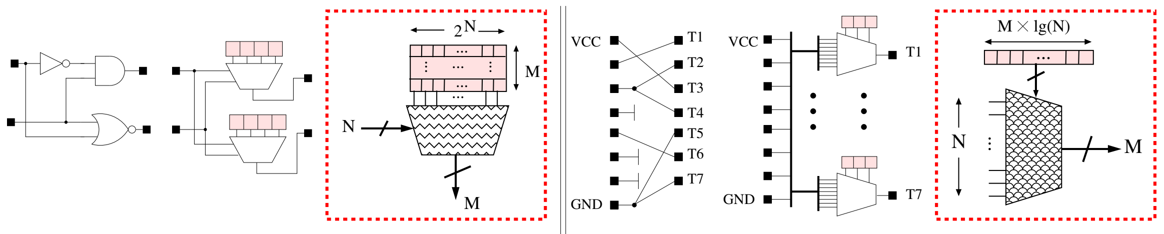
Fig. 2(c) shows a last example of obfuscation scheme using reconfigurable logic insertion to allow the designer to withhold any general function block in the design (Baumgarten et al., 2010) (Khaleghi et al., 2015). Here, various functional parts of the original design are replaced by Look-Up Tables (LUTs). These LUTs consist of memory cells which will eventually be configured to be the truth-table of the functions that were replaced, in the post-manufacturing stage, to restore the correct functionality of the chips.

1.2.1 MUX-based implementation of obfuscation schemes

A key insight derived from a broad examination across various obfuscation approaches is that the memory cells (holding the "key") on chip usually provide controls to select, among multiple possibilities, the "correct" configuration, for an "unlocked" chip. The configuration possibilities could take various forms, such as the truth-table of a logic function 2(c), or how wires are connected together 2(b). Theoretically, with k selection bits, one can control / select among 2^k choices, and this is the functionality of the fundamental digital logic building block of multiplexers (MUXs).

Based on the way of how a MUX is used in the context of various hardware obfuscation schemes, two categories of MUX-based implementation of obfuscation schemes exist, depending on where the key cells will be located. We therefore denote them as Content Obfuscator, used for implementing the obfuscation scheme shown in 2(c), and Wire Obfuscator, used for implementing the obfuscation scheme shown in 2(b).

Fig. 3(a) shows an example (of using LUT to obfuscate a 2-input / 2-output function) and the general form of a *Content Obfuscator*, that can be used to implement any general ways of using LUTs to replace an N-input / M-output logic function, including the key-gate based and LUT-based obfuscation



(a) Content Obfuscator (b) Wire Obfuscator

Figure 3: Two MUX-based obfuscation implementations with examples and general forms

approaches (Roy and Koushanfar, 2008) (Rajendran et al., 2012b) (Bao and Wang, 2014) (Rajendran et al., 2012a) (Baumgarten et al., 2010) (Khaleghi et al., 2015). The peripheral circuit is then connected to the MUX via the N select lines and M outputs, while the total number of $M \times 2^N$ bit key cells (essentially forming the truth-table of a LUT) are located at the input lines of the MUX. The size (and therefore hardware costs) of a Content Obfuscator grows exponentially to its input width N.

Fig. 3(b) shows an example (of configuring interconnects between 8 inputs and 7 outputs) and the general form of a *Wire Obfuscator*, that can be used to implement the general interconnect mapping configurations between N and M ends. To choose from N potential input bits, $lg(N)$ cells are needed at the select lines of a MUX. When M output bits are needed to be chosen from the N input bits, a total of $M \times lg(N)$ key cells are needed, positioned at the select lines. This is a typical "multiplexing" model to provide "configurable switching" control ability, and can be used to model the wiring scramble obfuscation schemes (Zamanzadeh and Jahanian, 2013). When using MUXes to implement the mapping, the key cell values are located on the select lines of the MUX, and the peripheral circuits are connected to the MUX via the N input lines and M output lines. The number of key cells of a Wire Obfuscator grows in logarithm scale to the input width N, while the MUX overhead (in terms of hardware cost) grows linearly to the input / output width (N, M).

1.3 Attack Models and Assumptions

Despite the variety in their approaches, a common strategy among the existing hardware obfuscation schemes is to replace a certain piece of design information by a configurable module, eventually implemented with (tamper-resistant) memory cells on a chip. The post-manufacturing stage of configuring such memory cells makes it possible for the designer to both conceal / withhold some crucial design information from the potential attackers, as well as unlock fabricated chips. While most obfuscation approaches try to ensure that the required effort for an attacker to obtain the correct key is computationally hard, significant challenges still remain, as most approaches, as well as the selection of obfuscation targets, are ad-hoc based (Roy and Koushanfar, 2008) (Rajendran et al., 2012b) (Bao and Wang, 2014) (Rajendran et al., 2012a) (Alkabani and Koushanfar, 2007) (Baumgarten et al., 2010) (Zamanzadeh and Jahanian, 2013), making them susceptible to various attacks.

Assuming the strongest type of attackers, in the position of a manufacturer, we categorize the attacks into two types: 1) algorithmic attack on the partially available obfuscated design file, and 2) physical attack on an unlocked chip, as is illustrated in Fig. 4.

Rather than focusing on a specific set of detailed attack types and assumptions, we focus on an abstract level of attack taxonomy to provide a basis for defense mechanisms. For example, a number of different techniques can be applied in attempt to read out the content of the key memory cells on-chip, such as side-channel attacks or UV radiation, yet instead of proposing schemes to deal with each form

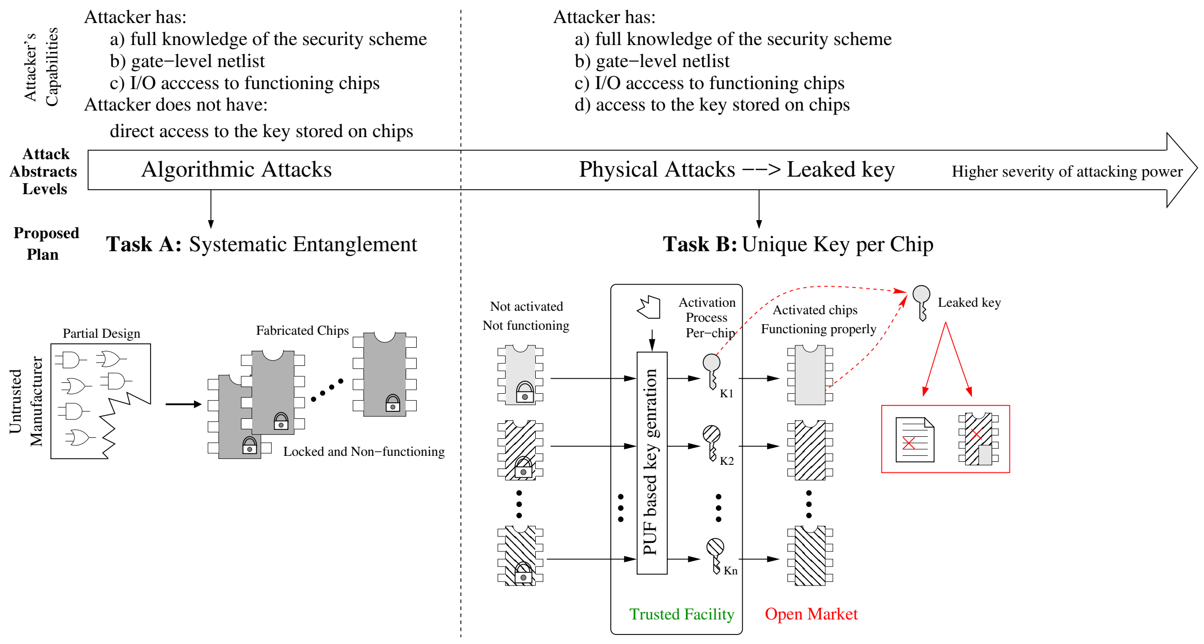

Attacker's Capabilities
Attacker has:
a) full knowledge of the security scheme
b) gate−level netlist
c) I/O acccess to functioning chips
Attacker does not have:
direct access to the key stored on chips
Attacker has:
a) full knowledge of the security scheme
b) gate−level netlist
c) I/O acccess to functioning chips
d) access to the key stored on chips
Attack Abstracts Levels
Algorithmic Attacks
Physical Attacks −−> Leaked key
Higher severity of attacking power
Proposed Plan
Task A: Systematic Entanglement
Task B: Unique Key per Chip
Untrusted Manufacturer
Partial Design
Fabricated Chips
Locked and Non−functioning
Not activated
Not functioning
Activation Process Per−chip
PUF based key genration
K1
K2
Kn
Activated chips
Functioning properly
Leaked key
Trusted Facility
Open Market

Figure 4: An overview of attack models, assumptions, and the proposed defensive mechanisms

of the attacks specifically, based on their physical access mechanisms, the proposed research will focus on how to ensure that an assumed leaked key will render useless for the attacker.

For algorithmic attacks, we assume that the attacker possesses: a) complete knowledge of the security scheme, b) gate-level netlist of the design, except for the obfuscation information, c) input / output access to some legally unlocked chips. The attacker is also assumed to possess powerful simulation tools and computation abilities. However, here the attacker is assumed to *not* have direct access to the keys on-chip. These assumptions are commonly recognized and adopted by many of the IC protection schemes, assuming a tamper-resisitant on-chip memory module to be used for the keys (Roy and Koushanfar, 2008),(Rajendran et al., 2012b), (Chakraborty and Bhunia, 2008), (Chakraborty and Bhunia, 2009), (Baumgarten et al., 2010) (Khaleghi et al., 2015) (Zamanzadeh and Jahanian, 2013).

For the next stage of physical access attack category, the attacker is assumed to have everything from above in the algorithmic attack category. In addition, the attacker can gain *full* content of the key cells of a chip.

1.3.0.1 Algorithmic attacks on obfuscated design files

Via *brute-force attack*, as is shown to a Content Obfuscator in Fig. 5(a), an attacker needs to solve for all of the $2^4 = 16$ possible patterns of the key, which is exponential to the key size and input size, and employ a verification process for every potential key pattern. The verification process involves, in the worst case scenario, applying all the possible input combinations (from 000 to 111 of $2^3 = 8$ in this example), to both the simulated design (with the potential key), and the unlocked chip, for comparison. A mismatch will indicate the key pattern to be wrong, while a matching takes all the input patterns to validate a correct key pattern. Therefore, any obfuscation scheme with a Content Obfuscator is generally effective against brute-force attacks, by imposing a prohibitively expensive computational cost (exponential to both the key size and the input size) to the attacker.

Unfortunately, an attacker equipped with a partially available design and an unlocked chip can resort to the more effective *algorithmic attack*, by focusing on **one key cell at a time**. The main idea is to use an input combination that can successfully “isolate” one target key cell, and propagate its value towards the outputs. In the example shown in Fig. 5(a), if an input combination can “activate” the address of key cell k_1, and propagate the value cell to the primary output, then the value of k_1 can be obtained by observing the output of the legally unlocked chip. If each key cell can be cracked this way, then the

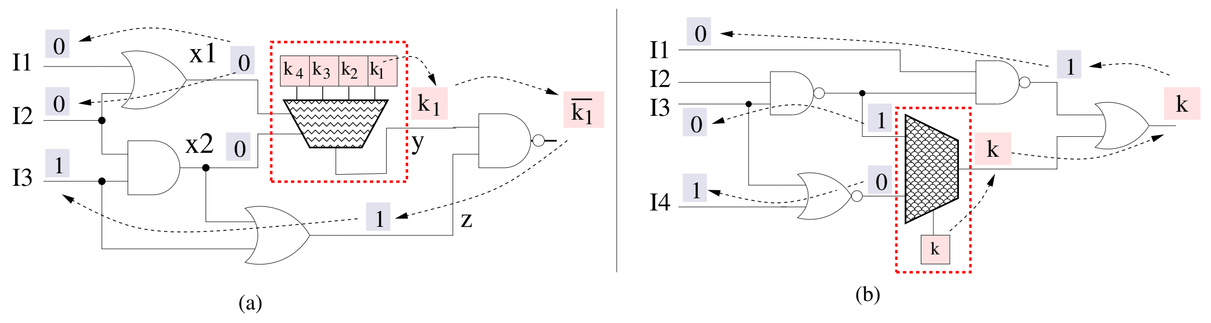


Figure 5: Examples of algorithmic attacks on: (a) a Content Obfuscator (b) a Wire Obfuscator

number of trials for the attacker will be reduced to the size of the key - a complete collapse from the brute-force attack complexity.

Fig. 5(a) shows the process of algorithmic attack for finding such an input combination to crack k_1. In order to activate the address of k_1, the attacker needs to use an input combination that makes $x1 = x2 = 0$. Based on the available design, it can be deduced that there are two input combinations for $(I1, I2, I3)$ that can satisfy this condition: (0,0,0) or (0,0,1), as both patterns will select the value of k_1 to appear on wire y. However, to ensure that k_1 can be propagated all the way to the primary output, (0,0,0) does not qualify, because the signal at y will eventually be blocked by $z = 0$. On the other hand, input combination (0,0,1) can successfully reveal $\bar{k_1}$ to the primary output. In general, using algorithmic attacks to solve for a particular key cell of a Content Obfuscator, one needs to solve for the primary input combination, such that: 1) the address for the key cell is selected, and 2) the value (or its negation) of the cell can be propagated to one of the primary outputs.

A similar attack can be performed on a Wire Obfuscator, shown in Fig. 5(b). In this example, since the key is connected to the selector of the MUX, there is no direct way to send it to the output of the MUX. However, the attacker can try to send a pair of opposite values (1 and 0) to the input lines of the MUX, and observe the value on the output to deduce the bit at the selection line. One way to achieve this in the example shown in Fig. 5(b) is by making $I3 = 0$ and $I4 = 1$, and then set $I1 = 0$ to drive the value of k to the output end.

Such an algorithmic attack has been shown to be equivalent to the "Automatic Test Pattern Generation (ATPG)" problem, which is a classical NP-Complete problem in the domain of digital testing (Abramovici et al., 1990) [1]. Due to its NP-Complete complexity, sometimes it is assumed that algorithmic attack poses a high cost for the attackers.

However, while it is true that each run of the the algorithmic attack is either NP-Complete or NP-Hard (Abramovici et al., 1990) (Rajendran et al., 2012b) (Erb et al., 2013), our preliminary research data verify that it is dangerous for a protection scheme to rely solely on the hardness of such 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 (Khaleghi et al., 2015). The need of systematically introduced entanglement against algorithmic attacks is shown in Fig. 12: To perform algorithmic attacks on content obfuscator, while some of the worst cases could take up to 250ms to solve one key cell, the median runtime of all the benchmarks is as little as 27.7ms. Furthermore, due to its important application in chip testing, many

[1] ATPG by itself is concerned with finding an input pattern that can distinguish between the correct circuit and a faulty circuit, assuming a fault occurring at a particular location.

powerful tools and heuristic algorithms are available to tackle the ATPG problem effectively. For instance, a SAT-based algorithmic attack has been shown to be successful in retrieving the full / partial key from a fabricated chip (Subramanyan et al., 2015) efficiently.

1.3.0.2 Physical attacks on unlocked chips

The algorithmic attack assumes the attacker to have access to the input/output of an unlocked chip, but no access to the key memory cells. Such an assumption is held by most obfuscation schemes, emphasizing that the memory cells where the key is stored on chip can be made tamper-resistent. However, physical access attacks do pose a powerful threat when the attackers are equipped with strong tools and abundant funds (Joye, 2009) (Stanojlovic and Petkovic, 2010) (Samyde et al., 2002) (Rakers et al., 2001) (Tiri et al., 2002) (Moore et al., 2003) (Iyengar et al., 2016) (Rostami et al., 2013) (Skorobogatov, 2009) (Skorobogatov, 2010). Even if only a portion of all the key cells are gained, in a statistical way, such information could still be used to aid the algorithmic attack to accelerate the cracking of the rest of the key tremendously. Even though such physical attacks are harder to carry out, and their results are probabilistic rather than deterministic, their existence has to be taken into account and modeled with their strongest form, to provide a solid ground of defense mechanisms. For the proposed research, we categorize the physical attack approaches based on *what* they are able to achieve, in the obfuscation paradigm, rather than *how* they are carried out.

Write-in of Key cells: A *write-in* attack is defined as for an attacker to set the value of some key cells of an unlocked chip, independent of the ability to read out their values. This is possible, as there needs to be a mechanism for the designer to write in the key into all the chips in the post-manufacturing stage. The same mechanism to write-in might be exposed to the attacker, even if the designer tries to remove the channel after the unlock process, because strong invasive schemes are possible from a manufacturer, such as a hidden write-access at the manufacturing stage, or via advanced tools such as radiation, etc. (Skorobogatov, 2009) (Skorobogatov, 2010).

Such write-in access can be used to efficiently help an algorithmic attack: an attacker can try out a specific value (say 1) by writing it into the key cell of an unlocked chip A, and then test it out for a number of input combinations and against another unlocked chip B with the correct key. During this process, a single mismatch can verify that the trial value (1 for example) of the target key cell is wrong, and therefore the true key value should be the opposite (in this case 0). In the case that no mismatches are found for a while, the inserted value is likely to be the actual content of the target cell, and the attacker can write in the opposite value into the same cell to seek for a mismatch as verification. This way, the attacker has a large probability of cracking a portion of the key cells within a limited number of trials.

Read-Out of Key cells: Undoubtedly, the most desirable ability from an attacker's point of view is to read out the values of key cells, from an unlocked chip. This is possible for powerful attackers - to probe and read out the content of the key memory directly, with approaches such as Side-Chanel attacks despite being expensive and inexact, to at least read out a portion of the key (Joye, 2009) (Stanojlovic and Petkovic, 2010) (Samyde et al., 2002) (Rakers et al., 2001) (Tiri et al., 2002) (Moore et al., 2003) (Iyengar et al., 2016) (Rostami et al., 2013).

In the extreme case of a completely leaked key, assuming a common key is used to obfuscate all the chips, the attacker achieves the ultimate goal of cracking the obfuscation scheme: an untrusted manufacturer can then fabricate more chips and directly use the leaked key to unlock them for sale.

Overall, regardless of the various ways that an attacker might have to read out the content of the protected memory on chip, such a capability needs to be considered in a worst-case scenario, to construct a provably strong defensive mechanism.

1.4 Organization

The organization of this dissertation is as follows. Chapter 2 deals with the algorithmic attacks by proposing a novel protection scheme, called Entanglement: 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.

Chapter 3 lays out the foundation for preventing the physical attacks by proposing the idea of "group formation" to exploit the nano-scale analog disorders of devices for making "strong" Physically Unclonable Functions (PUFs) . PUFs are an emerging technology that could play the key roles in various security applications. Depending upon the size of its truth-table, i.e., the search space for an attacker to fully specify its behavior, a PUF can be categorized as either "weak" or "strong". This chapter presents a scheme for making a strong PUF based on Spin-Transfer Torque Magnetic RAM (STT-MRAM), an emerging nano-electronic memory device. In the end, this chapter sheds light on how to make a strong PUF in general, by extending the idea of group formation beyond the STT-MRAM devices.

Chapter 4 proposes a *strong* PUF-based hardware obfuscation scheme to effectively prevent IC piracy even in the case of a leaked key from some activated chip. To ensure that each chip has a unique key, PUFs have been proposed to be integrated with hardware obfuscation. Such a paradigm is constrained to use *weak* PUFs, because, to uniquely set the key (the content of the configurable module) for each chip, the designer needs to fully characterize the PUFs for all the chips. In this chapter, we argue that a powerful attacker in the position of a manufacturer can fully characterize all the weak PUFs, and use any leaked key to break the obfuscation framework. This chapter proposes to employ *strong* PUFs (with *huge* search space of truth-table) into the obfuscation framework, against such an attacker/manufacturer. While it is impossible for an attacker to fully characterize the strong PUFs, the main challenge becomes ensuring that the designer does not need to bear the burden of fully characterizing the strong PUFs to generate a unique key per-chip. This is achieved by employing an *Obfuscator* block into the design, which enables the designer to select an arbitrarily subset of the strong PUF to work, while guaranteeing that the architecture does not reveal to the attacker/manufacturer of the choices made by the designer.

Chapter 5 concludes this dissertation.

CHAPTER 2

HARDWARE OBFUSCATION THROUGH ENTANGLEMENT AGAINST ALGORITHMIC ATTACKS

2.1 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 (Roy and Koushanfar, 2008). 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* (Roy and Koushanfar, 2008), (Torrance and James, 2011): 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; either by direct access from the IC design or by reverse engineering of the layout, mask, or a manufactured IC. 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 (Baumgarten et al., 2010), (Zamanzadeh and Jahanian, 2013). 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.

The proposed work in this chapter 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.

2.2 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 (Kahng et al., 1998a) (Koushanfar et al., 2005) (Kahng et al., 1998b) (Lach et al., 1998), (Rostami et al., 2013). *Fingerprinting1* techniques work by embedding both the designer's watermark and the buyer's signature in the design. Not only can the designer claim the ownership of a design, but it can also reveal the source of piracy by retrieving the buyer's signature (Caldwell et al., 2004) (Holcomb et al., 2009) (Ruhrmair et al., 1011). Such schemes can passively provide mechanisms for detection of illegal copies, yet cannot *prevent* reverse engineering or IC Piracy from occurring.

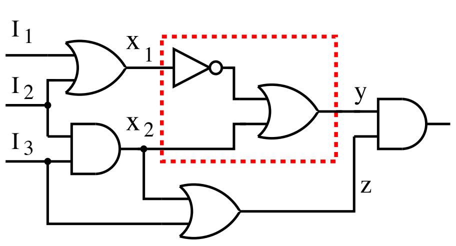
Obfuscation-based approaches, on the other hand, aim at "hiding" the design from potential attackers with extra obfuscating hardware, so that no manufactured IC can function correctly, unless being activated by its designer. Since the designer is the only one who knows the correct key, it can control the number of functioning IC's, and prevent untrusted manufacturer from conducting IC piracy (Roy and Koushanfar, 2008),(Rajendran et al., 2012b), (Alkabani and Koushanfar, 2007), (Chakraborty and Bhunia, 2008) (Chakraborty and Bhunia, 2009). 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 (Baumgarten et al., 2010), (Bao and Wang, 2014). 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 (Zamanzadeh and Jahanian, 2013).

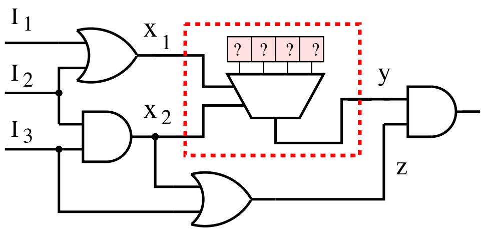
In fact, the obfuscation-based techniques can be covered in the withheld-based framework, making it easier to use the latter to develop theoretical foundation for trustworthy schemes. Furthermore, as opposed to the obfuscation-based approaches where the entire (obfuscated) design is made available the 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.

2.3 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) Design with partially withheld piece



(b) Chip model

Figure 6: An example, where a part of a design (including 2 inputs and 1 output) is replaced with a LUT on chip.

2.3.1 Withholding a single-output function

Suppose the circuit in Fig. 6(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. 6(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 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 (Abramovici et al., 1990). 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.

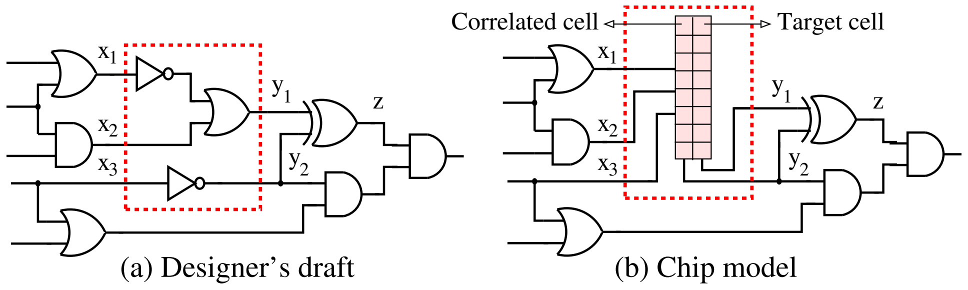


Figure 7: Withholding a multi-output function to elevate each attack from NP-Complete to NP-Hard complexity.

2.3.2 Withholding a multiple-output function

Fig. 7(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. 7(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. 7(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 (Erb et al., 2013), 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 (Erb et al., 2013).

2.3.3 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 (Fujiwara and Toida, 1982). 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 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.

2.4 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

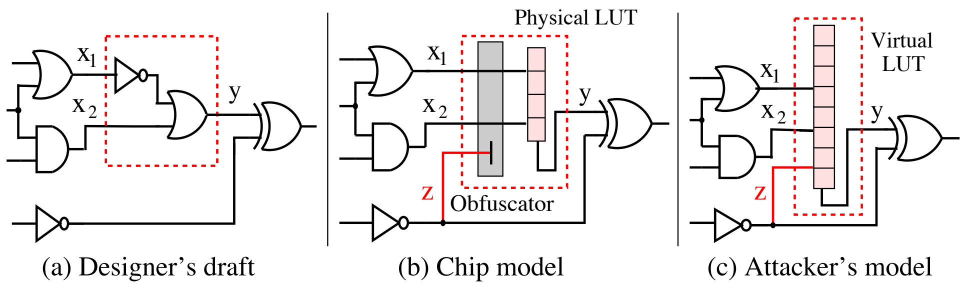


Figure 8: External Entanglement: a) the original withheld function; b) designer's cost (LUT size) kept low with the help of an Obfuscator; c) the attacker has to solve a much larger number of cells, due to the entanglement.

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.

2.4.1 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

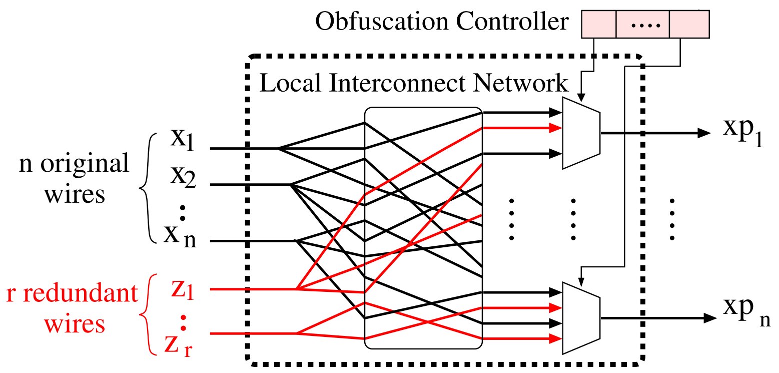


Figure 9: General structure of a programmable Obfuscator.

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. 8 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. 8(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. 8(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. 9 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.

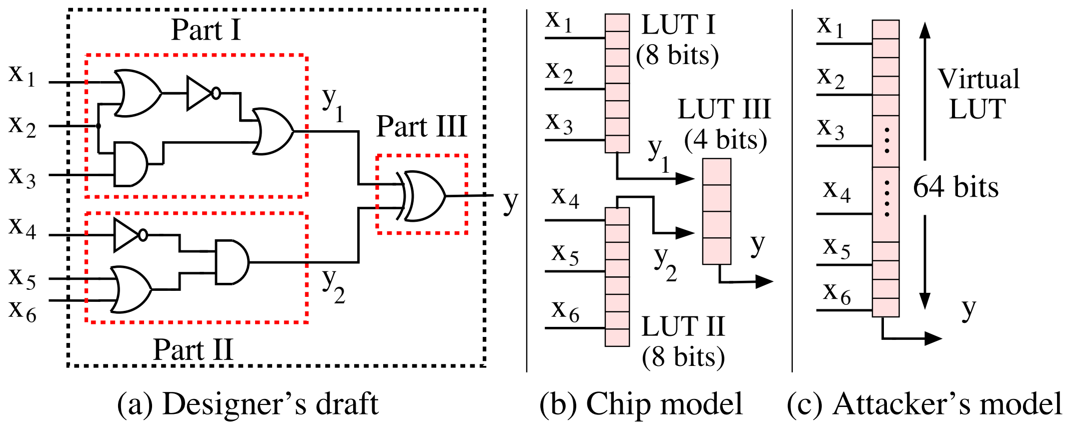


Figure 10: Internal Entanglement: a) a large withheld piece is decomposed into 3 smaller pieces; b) Using three interlocking small LUT's to shrink the hardware cost; c) the large virtual LUT remained for attacker to solve.

2.4.2 Internal entanglement

Fig. 10(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 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. 10(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. 10(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

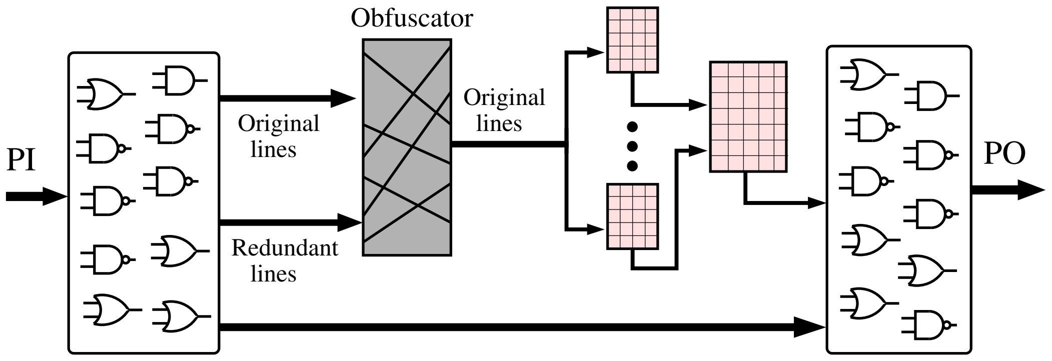


Figure 11: Applying both External Entanglement and Internal Entanglement.

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.

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. 11 shows a schematic design, where both entanglement techniques are combined together.

2.5 Evaluation

The effectiveness of the proposed scheme is analyzed using the ISCAS-85 combinational benchmarks. Attacks are simulated using the Atalanta ATPG tool (Lee and Ha, 1991).

Fig. 12 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

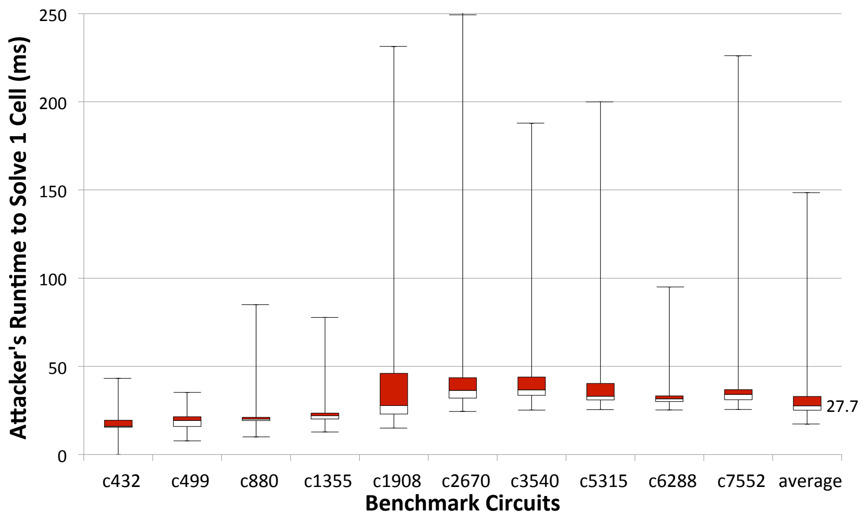


Figure 12: The minimum, first quartile, median, third quartile, and maximum runtime for an attacker to solve an input pattern for each cell, in the case of withholding one piece of design with a single-output (NP-Complete complexity) over all possible single-output functions.

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. 13 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 for the attacker. The time scale (10

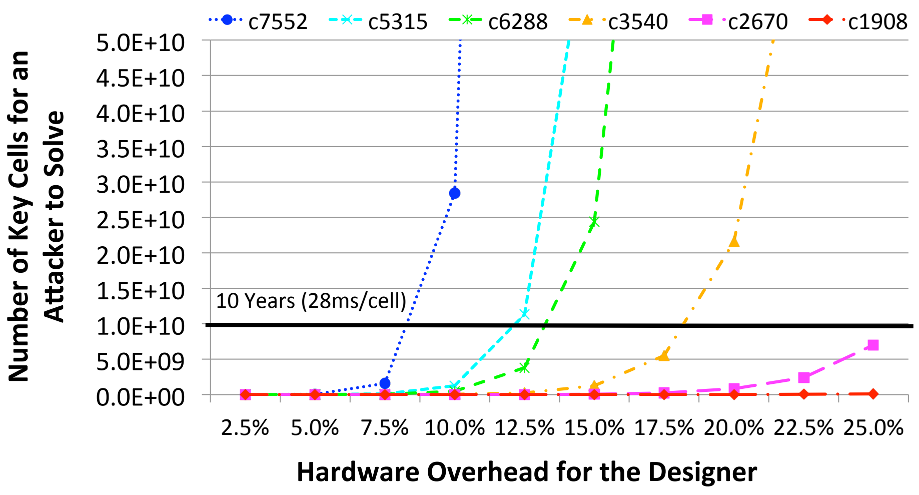


Figure 13: Attacking cost vs. Hardware cost for the Internal Entanglement scheme.

years line) is calculated using the median time obtained in Fig. 12 (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. 14, which shows the same data in a logarithmic scale.

Fig. 14 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.

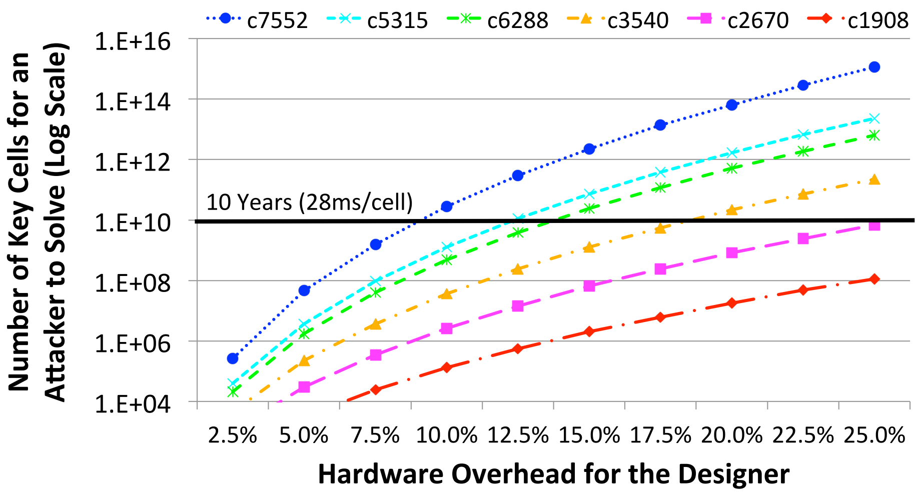


Figure 14: Attacking cost **(Logarithmic Scale)** vs. hardware overhead for the Internal Entanglement scheme.

2.6 Conclusions

Two Entanglement schemes are proposed in this chapter, 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 chapter 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 (Rostami et al., 2013), (Stanojlovic and Petkovic, 2010) (Joye, 2009). 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.

CHAPTER 3

A NEW WAY OF CONSTRUCTING STRONG PUFS WITH STT-MRAM AS A CASE STUDY

Parts of this chapter have been presented in (Khaleghi et al., 2016). Copyright © 2016, IEEE.

3.1 Introduction

While classical and modern cryptography schemes can effectively resist against powerful attacks, they all rely on the concept of a secret key. It is usually assumed that such a key can be securely stored in physical devices, perhaps inside some highly secure, tamper-resistant memories. In practice, however, powerful invasive and side-channel attacks can easily extract the key information form protected memories (Herder et al., 2014) (Rostami et al., 2013)(Joye, 2009)(Stanojlovic and Petkovic, 2010).

Physically Unclonable Functions (PUFs) are an emerging technology that could play the key roles in various security applications. Basically, PUFs can offer a unique key for every chip by deriving it from some noisy physical characteristic of the chip. Particularly, one main advantage of PUFs is that they do not require the key to be explicitly stored on chip; instead, they provide a *challenge-response* mechanism via physical interaction, making them harder to crack for a variety of powerful attacks (Herder et al., 2014) (Ruhrmair et al., 2010). For example, any attack on a PUF device must be attempted while the chip is powered on; otherwise, no information can be gained. In addition, invasive attacks, which are known to be powerful for extracting the key from a digital memory, usually affect the physical characteristics of the IC, based on which, a PUF is built. Therefore, invasive attacks may render useless, as they would essentially change (destroy) a PUF device (Devadas et al., 2008). Furthermore, since PUFs are based on nano-scale structural disorders, they cannot be cloned physically, even by the same manufacturing process.

Each PUF can be essentially seen as a function, providing a unique way of mapping the challenges into the responses. Depending upon the size of its truth-table, i.e., the search space for an attacker to fully specify its behavior, each PUF can be categorized as either "weak" or "strong". Weak PUFs offer a limited search space, polynomial with respect to the number of their building components. Strong PUFs, on the other hand, offer a huge search space, exponential with respect to the number of their components. This makes them suitable for a wider range of security applications (Herder et al., 2014).

Spin-Transfer Torque Magnetic RAM (STT-MRAM) is an emerging Non-Volatile Memory (NVM) that can offer low power consumption, and high scalability (Wolf et al., 2010). Process variations are increased due to technology scaling of NVM devices, making them a good candidate for PUFs. The work in (Zhang et al., 2014) proposes a weak PUF based on STT-MRAM, which is capable of producing response bits with desirable randomness and reliability.

It is usually assumed that the type of a PUF, whether strong or weak, is inherently determined by its architecture and the kind of nano-scale analog disorders, based upon which a PUF is built. In this chapter, we will show that this assumption is not necessarily true, by presenting a scheme to construct a strong PUF based on STT-MRAM devices. To achieve a huge search space, we propose the idea of "group formation" to exploit the nano-scale analog disorders of STT-MRAM devices. Furthermore, we will discuss the necessary conditions for making a strong PUF in general by extending the idea of "group formation". This chapter also applies the generalized idea of "group formation" to a popular CMOS-based weak PUF to form a strong one.

3.2 Preliminaries

In this section, we briefly explain the basics of PUF devices, and define the two primary PUF types: "weak" and "strong".

3.2.1 General concepts of a PUF

Due to the nano-scale structural disorders, occurring during the fabrication phase of IC production, each chip is slightly different from the others, made with the same fabrication process. A PUF is a physical system that presents unclonability by exploiting these slight variations. A PUF can be stimulated with external inputs, called *challenges*, upon which it reacts with corresponding outputs, called *responses*. Therefore, every PUF implements a unique way of mapping challenges to responses for a specific IC. Since exact control over the manufacturing process is impossible, it is infeasible to build identical PUFs with the same Challenge-Response Pairs (CRPs). Consequently, it is also presumably impossible to predict the behavior of a PUF, unless by testing its all possible CRPs. Furthermore, PUFs can be usually implemented with a very small hardware investment, proportional to the number of challenge and response bits. Therefore, from a hardware perspective, PUFs are easy to build, but hard to duplicate; and from a software perspective, PUFs are easy to evaluate, but hard to predict.

In general, the potential applications of a PUF depends heavily on the number of CRPs that it can offer (Herder et al., 2014).

3.2.1.1 Weak PUFs

Some PUFs offer a **limited** number of CRPs, ranging from one (in the most extreme case) to polynomial-sized with respect to the number of their building components.

For example, the power-on state of an SRAM cell constitutes a weak PUF (Holcomb et al., 2009). Each SRAM cell has a tendency towards logic 1 or 0, due to process variations. Basically, every SRAM cell has two identical positive feedback at the device level, forcing the cell to either of the states during a write operation. If no write is performed at power-on, the more powerful feedback (due to process variations) will force the cell into its associated state. As a result, the initial power-on state of each SRAM cell will be either 1 or 0, making each cell a weak PUF with one CRP: the challenge is the powering on the cell, and the response is the initial state of the SRAM cell. Note that employing more SRAM cells would only increase the number of response bits, not that of CRPs.

Due to their limited number of CRPs, weak PUFs are mostly used for key-generation purposes in cryptography, where a few unique keys must be generated for every chip (Herder et al., 2014) (Sush and Devadas, 2007). However, since the entire truth-table of a weak PUF can be obtained in polynomial time complexity, its CRPs must be kept secret to prevent the potential attackers from building up the entire truth-table of the PUF, and emulating its behavior.

3.2.1.2 Strong PUFs

Some PUFs, on the other hand, offer a **huge** number of CRPs, usually exponential to the number of their building components.

Figure 15 shows an example of a strong PUF, called Arbiter (Devadas et al., 2008). The actual propagation delay of each Multiplexer (MUX) differs slightly from the others due to process variations.

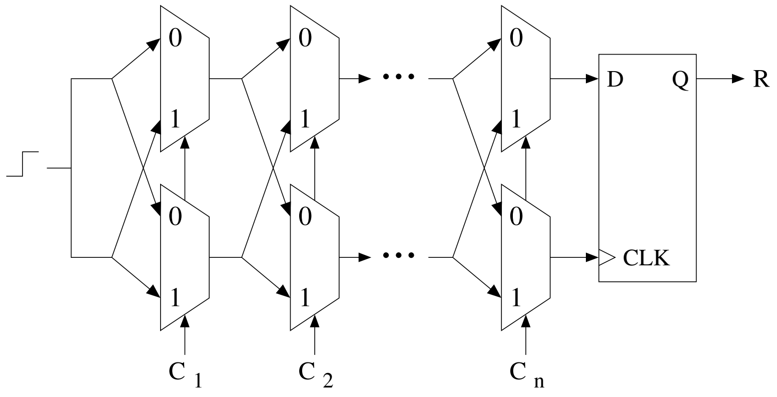


Figure 15: Arbiter PUF circuit: two delay paths are created based on the challenge (C_1 to C_n). Depending on the arrival times of the rising edge at the inputs of the D-FF (arbiter), the response (R) could become either “1” or “0”.

A signal transition, say from 0 to 1, will propagate through two different paths of MUXes, determined by the challenge (C_1 to C_n). Each challenge bit feeds in a set of two MUXes that are positioned vertically, so that the two selected paths would not share any of the MUXes. Depending on the order of arrivals of the rising edge at the terminals of the D-Flip Flop, the response bit (R) would be either 0 or 1. In this PUF, the challenges are the selection of any two complementary paths, a total of 2^n possible CRPs, where n is half the number of MUXes.

Due to its huge number of CRPs, the security of a strong PUF does not rely on keeping its CRPs secret, but rather on the fact that recovering the entire truth-table of the PUF in a reasonable time is infeasible. Consequently, they can be employed in a wider range of security applications, such as authentication and logic obfuscation (Herder et al., 2014).

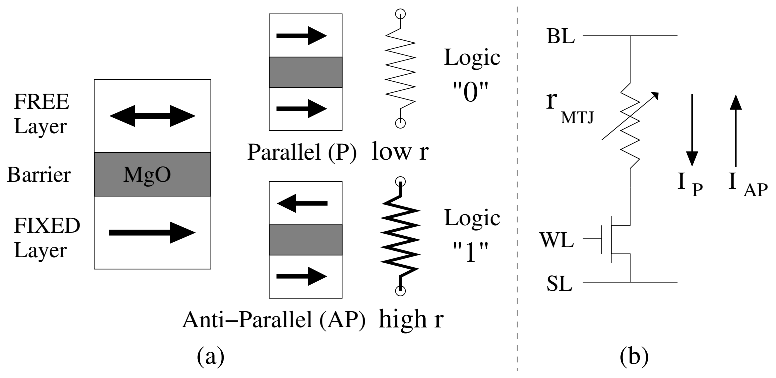


Figure 16: (a) MTJ device structure and its resistance model; (b) The complete structure of an STT-MRAM cell

3.2.2 STT-MRAM devices

Spin-Transfer Torque Magnetic RAM (STT-MRAM) is an emerging nano-electronic memory device that can offer significant performance improvement and power reduction (Wolf et al., 2010). Figure 16 shows the architecture of an STT-MRAM device. As it is shown in Figure 16(a), the storage part of the device is a Magnetic Tunnel Junction (MTJ), consisting of three layers: two ferromagnetic layers, separated by an insulating oxide layer. One of the ferromagnetic layers, called the “fixed layer”, has a fixed magnetization vector in any operating condition, while the other one, called the “free layer”, has a magnetization vector that is free to switch between two directions. Accordingly, the MTJ cell has two states, shown in Figure 16(a):

1. ***Parallel (P)***: when the magnetization directions of both ferromagnetic layers are the same, the MTJ cell has a **low** resistance, associated with **logic 0**.

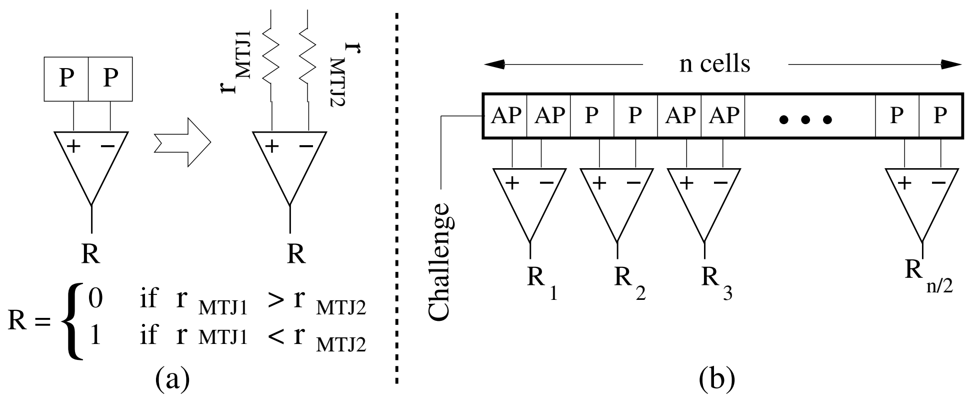

P P
+ −
R
r_{MTJ1}
r_{MTJ2}
R
R = { 0 if $r_{MTJ1} > r_{MTJ2}$; 1 if $r_{MTJ1} < r_{MTJ2}$
(a)
n cells
AP AP P P AP AP • • • P P
Challenge
R_1 R_2 R_3 $R_{n/2}$
(b)

Figure 17: An STT-MRAM based weak PUF: (a) The main idea: comparing the resistances of two identical cells with the same magnetization; (b) The complete architecture of the PUF.

2. ***Anti-Parallel (AP)***: when ferromagnetic layers have opposite magnetization directions, the MTJ cell has a **high** resistance, associated with **logic 1**.

Figure 16(b) shows a commonly used structure of an STT-MRAM cell. By allowing a current to flow through the device, this structure enables both Read/Write operations without relying on an external magnetic field. The magnetization direction of the free layer, which specifies the logic value of the cell, is determined by the direction of the current flow through the device. By comparing the resistance of a cell with a fixed reference resistance, the value of the cell, either 0 or 1 (*P* or *AP*) can be determined.

3.2.3 An STT-MRAM based weak PUF

Due to process variations, the equivalent resistance of an STT-MRAM cell in either of its states (*P* or *AP*) would be slightly different from those of the other cells (Zhang et al., 2014) (Wolf et al., 2010). Figure 17 depicts the architecture of a weak PUF based on the idea presented in (Zhang et al., 2014).

The main idea behind this PUF is to compare the resistance of two cells, set to the same state (either both to *P* or both to *AP*), as shown in Figure 17(a). Depending upon which cell has a slightly higher resistance, the response bit would be either 1 or 0.

Figure 17(b) shows the complete architecture of such a PUF: a memory composed of *n* STT-MRAM cells. In this PUF, the challenges are the pattern of magnetization (*P* or *AP*) of each pair of cells, and the responses are the outputs of the sense-amplifiers, comparing the resistances of adjacent cells. As it is shown in Figure 17(b), every pair of adjacent cells in such a PUF must be set to the same state; otherwise, the output of their corresponding sense-amplifier can be easily predicted by an attacker, as state *AP* has a higher resistance than state *P*.

In order to verify that such architecture is a weak PUF, one must consider the number of CRPs that can essentially reveal the entire truth-table of the PUF. Assuming that the outputs of all sense-amplifiers form a single response (consisting of $n/2$ bits), there exist $2^{n/2}$ CRPs for such a PUF. However, most of these CRPs are not independent from each other. In fact, the entire truth-table of this PUF can be obtained by examining the following two CRPs: the one with every pair set to *P*, and the one with every pair set to *AP*. All other CRPs can be predicted by referring to these two CRPs. This is due to the fact that the value of the response bit for each pair is independent from the states of other pairs. In other words, each pair of cells can only offer 2 valuable bits of information.

3.3 A Strong PUF Based on STT-MRAM

3.3.1 Motivation

Basically, every PUF is made by exploiting some noisy analog feature, presented at the implementation level of identically designed components. For example, the gate delays are the analog feature used in designing an Arbiter PUF; and the resistances of the MTJ cells are the analog feature in the given STT-MRAM based PUF example. These noisy analog features will eventually "collapse" into some digital bits, before they can be used as CRPs for security applications.

The infinite precisions of such analog features are inherently capable of offering an unlimited number of independent CRPs. The reason why some PUFs are "strong", while the others are "weak" has to do with how much precision of those analog features is exploited, before collapsing them into the digital domain. In the case of the weak PUF based on STT-MRAM cells, the resistance of a cell (an analog value) is compared with that of another cell to form a digital response (0 or 1). If instead of comparing two cells at a time, the resistances of a *group* of cells can be *combined*, and then compared with that of *another group*, a huge number of new CRPs can be introduced, exponential to the number of cells. We will provide two motivational examples to introduce the main elements that will be used in the proposed strong PUF.

1) Group Formation

The main idea of combining the resistances of a group of cells, before collapsing them into a digital signature, is to compare their overall resistance with that of another group. This is demonstrated with an example shown in Figure 18(a). This Figure shows a memory with 6 STT-MRAM cells, all of which are set to a **same fixed** magnetization (*P* in this case). The functionality of the *Group Formation Block* is to allow the formation of two 3-cell groups, so that the overall resistances of the two groups can be compared to form a response bit (signal *R)*. Based on the number of choices for the two groups, the total number of CRPs becomes $\frac{1}{2}\binom{6}{3} = 10$.

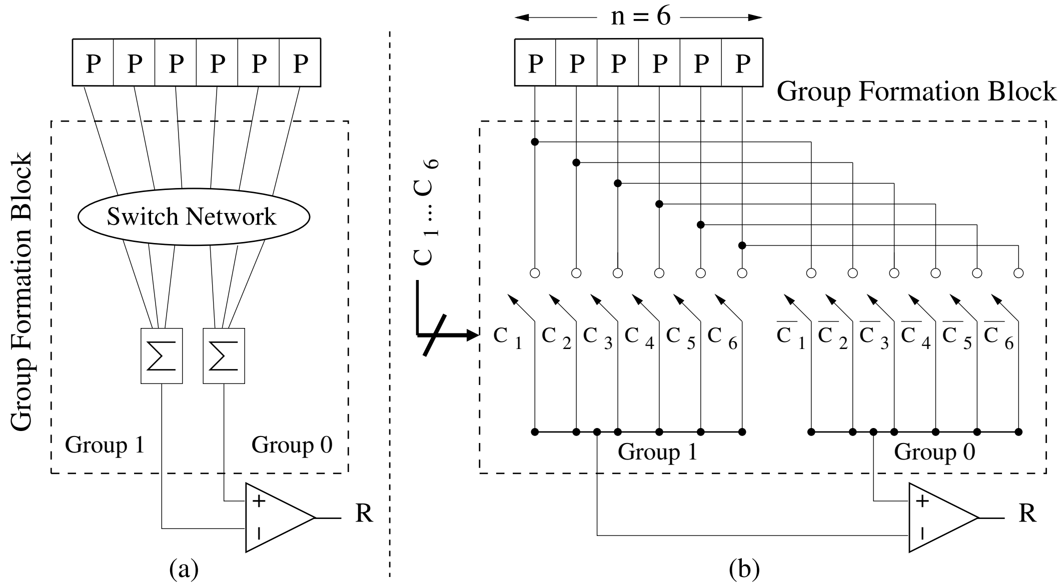


Figure 18: (a) The idea of forming 2 groups to increase the number of CRPs; (b) An example architecture for implementing the Group Formation Block for a PUF with 6 cells.

Figure 18(b) shows the architecture of the Group Formation Block. The two groups of cells (Group 1 and Group 0) are each connected to one port of the sense-amplifier. Each cell i can be selected to join either Group 1 or Group 0, by setting its corresponding challenge bit C_i. For example, if the first cell is to belong to Group 1 (connected to negative port of the sense-amplifier), then, its corresponding challenge bit must be set to 1 ($C_1 = 1$), so as to turn ON its corresponding switch in Group 1. It must be noted that a cell cannot be connected to both groups at the same time, due to the fact that the current division for each cell must be avoided. This is achieved by using 6 challenge bits (C_1 to C_6) to control 12 switches, in such a way that if the corresponding switch of any cell is ON for one group, the corresponding switch of the same cell for the other group is OFF, and vice versa.

2) Bit Pattern

Besides allowing the formation of two groups with multiple cells, another dimension to increase the number of CRPs is changing the bit pattern (magnetization vector) of cells. Even though such a dimension was used as the basis of the previous weak PUF, it was only exploited to a very limited extent. Under the group formation framework, changing the bit pattern of cells can significantly boost the number of CRPs.

Figure 19(a) shows an example of changing the bit patterns for a PUF with 6 STT-MRAM cells. In this example, each group has two cells in state P and one cell in state AP, which makes their resistance comparison able to serve as a PUF response bit. Figure 19(b) shows another valid CRP by changing both the groups and the bit pattern of the cells. Figure 19(c) is an example of an invalid CRP, because the total resistance of Group 1 is predictably larger than that of Group 0, as there are two cells in state AP in Group 1, while all the cells in Group 0 are in state P. Even though not all the CRPs are valid in such a PUF, there exists $\sum_{i=0}^{3} \binom{3}{i}^2 = 20$ valid CRPs for every single selection of two groups, enabled by changing the bit patterns of cells.

3.3.2 Architecture

The overall architecture of the proposed strong PUF, which is based on the ideas of group formation and changing the bit patterns, is depicted in Figure 20(a). Similar to the motivational examples, this architecture supports the combination of $n/2$ cells per group. By setting the challenge bits (C_1 to C_n), two groups of resistances would be connected to the two ports of the sense-amplifier. Then, the overall resistances of these two groups are compared to form a single response bit.

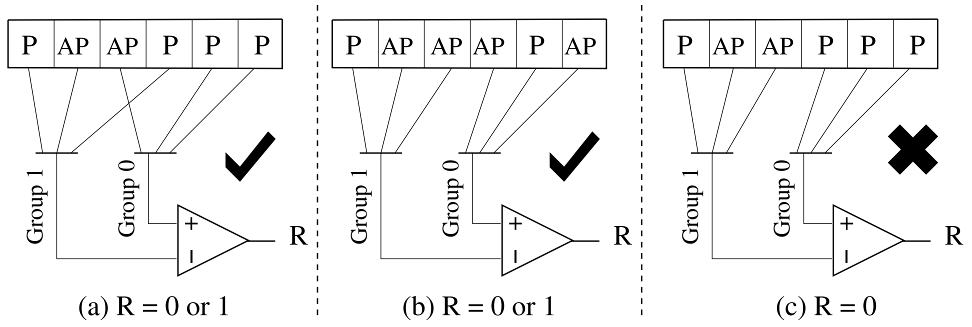


Figure 19: (a)(b) Examples of valid CRPs by changing the bit patterns and group formations; (c) An invalid CRP: Group 1 with two *AP*'s and one *P* has a higher resistance than Group 0 with three *P*'s.

Figure 20(b) shows an example of choosing a CRP for a PUF with 6 STT-MRAM cells. In this example, the cells 1, 2, and 4 are selected to form Group 1 (by setting $C_1 = C_2 = C_4 = 1$), and the other three cells are selected to form Group 0 (by setting $C_3 = C_5 = C_6 = 0$).

It must be noted that in such architecture, the overall resistance of each group is the *equivalent parallel resistance* of all the cells in that group. Consequently, the overall resistances of Group 1 (r_{G1}) and Group 0 (r_{G0}) in Figure 20(b) are determined by the following equations:

$$\frac{1}{r_{G1}} = \frac{1}{r_1^P} + \frac{1}{r_2^{AP}} + \frac{1}{r_4^P} \quad and \quad \frac{1}{r_{G0}} = \frac{1}{r_3^{AP}} + \frac{1}{r_5^P} + \frac{1}{r_6^P} \tag{3.1}$$

Since the overall resistances of the two groups must be equal in theory, the following constraints must be satisfied when selecting these groups:

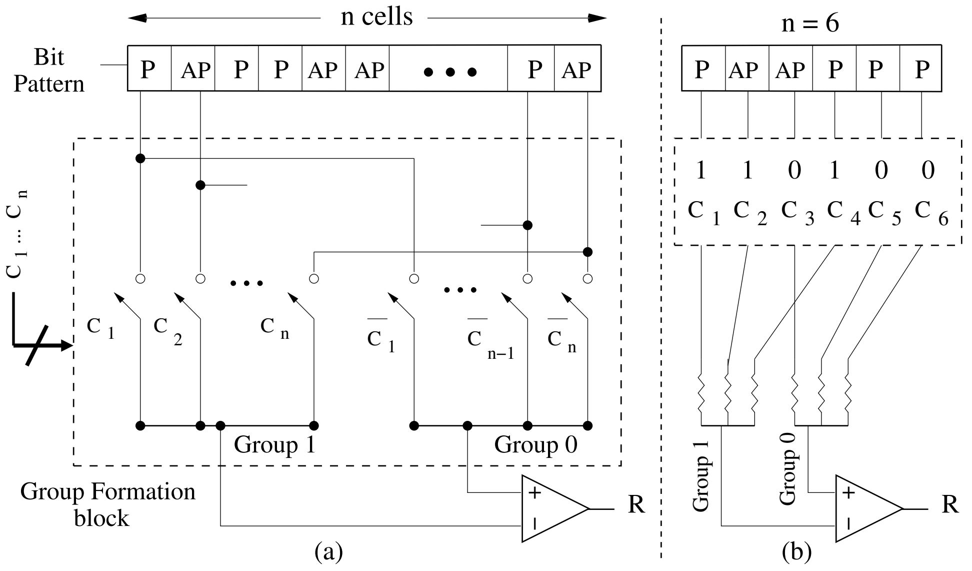


Figure 20: (a) The overall architecture of the proposed strong PUF; (b) An example for n=6: each group has two *P*'s and one *AP*.

1. the number of cells in each group must be equal to $n/2$; otherwise, the groups would be unbalanced and not useful. [1]

2. the number of cells in states *P* and *AP* must be equal in both groups; otherwise, the group with more cells in state *AP* would have a higher resistance, and the response can be predicted.

[1]It must be noted that the proposed architecture does not allow the formation of groups with equal number of cells, other than $n/2$ cells per group. This is to avoid the CRP information of the smaller groups to be used for determining that of the larger ones, which can be exploited by an attacker to characterize the PUF.

3.3.3 Analysis

To prove that the proposed PUF is in fact a strong one, we need to show that the number of CRPs is exponential (or larger) with respect to the number of elements in the PUF. As it was motivated in the previous section, each **challenge** in this PUF consists of 2 parts:

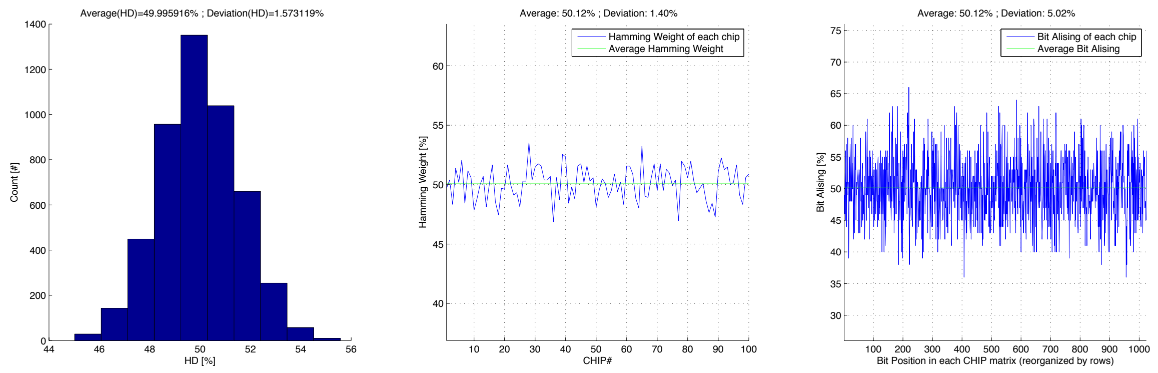
1. **Group Formation:** A part of challenge (C_1 to C_n) selects two groups of cells to be connected to the two ports of the sense-amplifier. Taking into account the 1st constraint above, there are $\frac{1}{2}\binom{n}{n/2}$ possible combinations of these groups for a PUF with n cells.
2. **Bit Pattern:** A part of challenge (initialization of STT-MRAM cells) specifies the state of each cell. Taking into account the 2nd constraint above, for a certain selection of the two groups (each with $n/2$ cells), the total number of possible combinations for the states are $\sum_{i=0}^{n/2}\binom{n/2}{i}^2$.

By putting together these two factors, the total number of CRPs for the given PUF is given as follows:

$$\frac{1}{2}\times\binom{n}{\frac{n}{2}}\times\sum_{i=0}^{n/2}\binom{\frac{n}{2}}{i}^2=\frac{1}{2}\binom{n}{\frac{n}{2}}^2 \tag{3.2}$$

It can be shown that the total number of CRPs, given in Eq. 3.2 grows faster than exponentially (factorial growth) with respect to the number of cells in the memory. As it was illustrated in the motivational examples, the information of none of these CRPs can be used to determine the responses of new challenges; thus, the proposed PUF is in fact a strong one.

In terms of hardware cost, only a single sense-amplifier is needed, compared to the $n/2$ sense-amplifiers in the weak PUF. $2n$ switches are required for implementing the Group Formation block. In



(a) Inter-chip Hamming Distance Distribution (b) Hamming Weight (c) Bit Aliasing

Figure 21: The quality of the proposed strong PUF with respect to three parameters: (a) Inter-chip Hamming distance: measuring the randomness among different chips; (b) Hamming weight: measuring the randomness among various bits of a same response within a same chip; (c) Bit aliasing: measuring the randomness for each response bit among various responses within a same chip.

terms of time overhead, the previous weak PUF can generate $n/2$ response bits at one cycle. For the proposed strong PUF, one bit is generated per clock cycle. Nevertheless, this time overhead is not an important issue in most of the security applications, especially due to the fact that at any time, one would only require to examine a few CRPs.

3.4 Evaluation

The purpose of this section is to verify the quality of the proposed strong PUF. Basically, a good PUF must offer maximum randomness of the responses within a chip (*intra-chip* uniqueness), and among different chips (*inter-chip* uniqueness). In other words, applying various challenges to a same PUF must generate random responses. Furthermore, applying the same challenges to various PUFs must result in random responses as well. Such uniqueness is usually measured by metrics such as *Hamming*

Distance, which shows the number of positions at which the two responses are different. In a fully random distribution (the ideal situation), the Hamming distance is equal to 50%.

Since the proposed strong PUF can generate one response bit per challenge, every 1024 response bits are treated as one response to measure intra-chip randomness. A total of 100 chips, each with 1000 CRPs are studied in this experiment. All simulations are performed in MATLAB by adopting the mathematical models of STT-MRAM devices from (Zhang et al., 2014).

Figure 21(a) shows the distribution of the inter-chip average Hamming distance for 100 chips. Basically, the average Hamming distance between every possible pair of chips is calculated by computing the Hamming distances of the same CRPs for every pair of chips. Then, the distribution of the average Hamming distance is plotted. As it can be seen from the graph, the median of this distribution is 49.99%, which shows a promising inter-chip randomness.

In order to evaluate the intra-chip randomness, two metrics are employed: 1) *Hamming Weight*, which measures the randomness of bits within the same response for a certain chip; and 2) *Bit Aliasing*, which measures the randomness of bits, placed at the same position among different responses for a certain chip. Figure 21(b) plots the Hamming weight of every chip, which is calculated by calculating the average Hamming weights of all possible pairs of responses within each chip. Figure 21(c) plots the bit aliasing of each chip, based on the bit positions, calculated for every possible pair of responses as well. As it can be seen from these graphs, the average values for both of these metrics are 50.12%, which verifies a promising randomness within each chip as well.

3.5 Strong PUF Beyond STT-MRAM

In this section, we will extend the idea of “group formation” beyond the STT-MRAM devices, and discuss the necessary conditions for making a strong PUF in general.

As it was discussed earlier, each PUF is built upon some noisy analog feature with infinite precision. To make a strong PUF, such infinite precision of those analog features must be exploited extensively before collapsing them into the digital domain, in such a way that the number of CRPs grows exponentially with respect to the building components of the PUF. There are two necessary conditions for making a strong PUF:

1) Device-level compatibility for group formation: The key idea for making a strong PUF is to see whether or not, the analog feature of a given component can be *combined* to from a group of components, *before* being collapsed into the digital domain. If so, the options of which components to be combined into groups will greatly increase the search space of the PUF. For example, the group formation is inherently supported by the analog feature of the STT-MRAM devices, the resistances of the MTJ cells.

However, such a combination of components is not always supported by the analog feature of the PUF device. For example, as is stated in section 3.2.1, the power-on state of a traditional SRAM cell is a weak PUF. In this case, the analog noisy feature is the two identical positive feedback at the device level, forcing the cell to either of the states during a write operation. In this example, the analog feature, which is not accessible at any design-level higher than the device level, does not support the necessitated feature as discussed to work in the group formation framework.

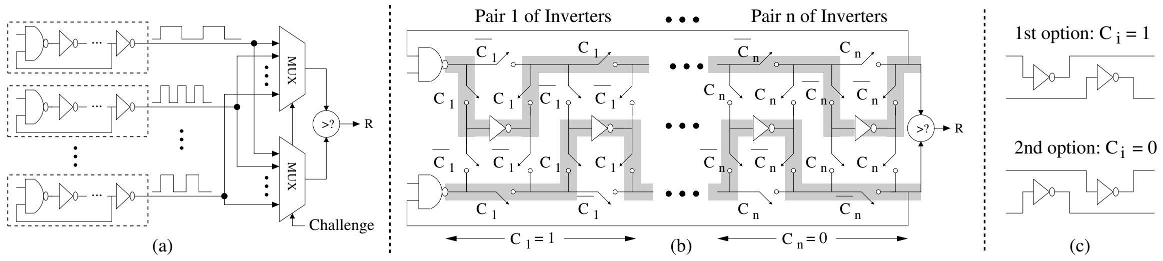


Figure 22: (a) RO-based weak PUF circuit: the frequencies of two ROs are compared to form the response bit *R*; (b) The proposed Strong PUF based on the idea of group formation: each Inverter in every pair belongs to one of the two ROs; The highlights correspond to the first and the last pair of Inverters for $C_1 = 1$ and $C_n = 0$; (c) The two options for each pair of Inverters ($1 \leq i \leq n$).

2) Architecture-level support to achieve exponentially large number of CRPs: Once the feasibility of combining the analog feature is determined, the same principle can be adopted in various architectures to boost the number of CRPs.

Next, we provide some insights on how to make a strong PUF in general based on the group formation scheme by presenting a case study of a popular CMOS-based PUF.

3.5.1 Case study of a RO-based strong PUF

Figure 22(a) shows an example of a popular weak PUF based on identical Ring-Oscillators (ROs), proposed in (Sush and Devadas, 2007). The actual frequency of each RO is slightly different from the other ones within the same chip. In such a PUF, the challenges are the selection of any two ROs, and the responses are either 1 or 0, depending on which of the two ROs has a slightly higher frequency. This PUF has a total of $\binom{n}{2}$ possible CRPs, which is polynomial with respect to the number of ROs.

As a matter of fact, not all of these CRPs are independent. For example, if RO A has a higher frequency than RO B, and B has a higher frequency than C, then, the response from comparing A and C can be predicted without actually examining it. Therefore, the full characterization of this PUF can be achieved by collecting the information for as low as $O(n \times log(n))$ CRPs, as such a characterization problem is equivalent to the classical sorting problem.

This results in a total of $\binom{n}{2}$ possible CRPs, where n is the number of ROs. In theory, there exist $n!$ possible distinct PUFs for this architecture. Thus, by choosing a large enough n, it would be unlikely for two different chips to have the exact same CRPs.

It can be seen that the analog feature in this PUF is the gate delays of the Inverters, which supports the idea of group formation at the device level, i.e., combining a number of Inverters can make ROs with slightly different frequencies. Figure 22(b) depicts the architecture of a RO-based *strong* PUF. Similar to the proposed strong PUF based on STT-MRAM devices, it guarantees the exponential number of CRPs by maintaining the following two conditions: 1) fixing the group size to be always equal to n (for $2n$ components); and 2) allowing each component to be in one of the two, bot not both, groups.

This architecture has n pairs of Inverters that play the key roles in obtaining the exponential number of ROs. As illustrated in Figure 22(c), the status of switches for every pair of Inverters are strongly correlated, so that if one Inverter belongs to one RO, the other Inverter in the pair belongs to the other RO (total of two options per pair). This is to ensure that each RO has exactly n Inverters (condition 1). Furthermore, each Inverter can belong to either of ROs, but it cannot belong to both of them at the same time (condition 2). As an example, the highlights in Figure 22(b) correspond to the first and the last pair of Inverters for $C_1 = 1$ and $C_n = 0$, respectively. Since there are two options for each pair of the Inverters, there exists a total of 2^n CRPs in this PUF, making this architecture a strong PUF.

In general, if the group formation is supported by the analog feature of a device, then, the implementation of the two above mentioned conditions in an architecture with $2n$ components can result in a total of 2^n CRPs, thus achieving a strong PUF. The first conditions essentially forbids the formation of groups with equal number of cells, other than n cells per group. Otherwise, an attacker can use the CRP information of the smaller groups to determine that of the larger ones. The second condition guarantees the exponential number of CRPs by allowing each component to belong to either of the two groups; thus doubling the search-space of the PUF for each additional component.

It can be seen that the proposed strong PUF requires additional hardware to implement the switches. However, it reduces the number of ROs in the weak version down to two instances. It also does not require the potentially huge MUXes, required in the weak version. In fact, depending on the number of ROs and their sizes for a certain weak PUF, the strong version could offer a huge hardware reduction.

3.6 Conclusions

In this chapter, we present a method for making a strong PUF based on STT-MRAM technology. Simulation results confirmed the effectiveness and uniqueness of the proposed strong PUF. We also discussed the possibility of making a strong PUF in general, by presenting the required conditions both at the device-level and the architecture-level. Even though the security of PUFs is still under investigation, and it is not clear which technology could be used in which applications, the proposed approach of group formation in this chapter is of a general framework that can be applied to a wide range of devices for building strong PUFs.

CHAPTER 4

INTEGRATING STRONG PUFS INTO THE HARDWARE OBFUSCATION FRAMEWORK AGAINST LEAKED KEYS

4.1 Introduction

The globalization of the semiconductor industry has raised serious concerns about the security of Integrated Circuits (ICs). Since IC designers no longer have full control over the manufacturing process, a design is prone to various hardware attacks from a malicious manufacturer. Particularly, a manufacturer can conduct *IC piracy* via producing unauthorized extra chips and/or stealing the information of a design through reverse engineering attempts (Rostami et al., 2013) (Koushanfar, 2012).

Hardware obfuscation schemes aim at preventing IC piracy attacks by enabling the designers to control the number of functioning chips through a post-fabrication activation process (Rostami et al., 2013) (Koushanfar, 2012). The key idea is to withhold a part of the design and replace it with a configurable module at the design stage, so that none of the manufactured chips would function properly without being "activated" by the designer (Chakraborty and Bhunia, 2008)(Alkabani and Koushanfar, 2007)(Rajendran et al., 2012b) (Baumgarten et al., 2010) (Zamanzadeh and Jahanian, 2013). Such a post-fabrication activation is achieved by securely restoring the withheld function back into the chips through specifying the content of the configurable modules. Afterwards, the chips are considered "unlocked" and can be made available to the open market. Without direct access to probe the securely stored *key* (the content of the configurable module) for the activated chips, an attacker cannot recover the entire design or overbuild illegal ICs.

Various types of hardware obfuscation approaches have been proposed in the literature. The combinational-based schemes work by inserting additional XOR gates with configurable bits into the design that must be set correctly to activate the chips (Rajendran et al., 2012b) (Koushanfar, 2012). The sequential-based approaches work by inserting additional “dummy” states into the Finite State Machine (FSM), so that the circuit functions properly only when a certain sequence of inputs (the key) is applied to the chips (Chakraborty and Bhunia, 2008)(Alkabani and Koushanfar, 2007). Permutation-based techniques propose to scramble the interconnect network of the design, so that only the correct key can configure the original interconnect network (Zamanzadeh and Jahanian, 2013). The withheld-based schemes work by replacing a part of the design (the key) with a Look-Up Table (LUT) that must be configured properly to activate the chips (Khaleghi et al., 2015) (Baumgarten et al., 2010).

The obfuscation schemes try to ensure that the required effort for an attacker to obtain the correct key is computationally infeasible (Rostami et al., 2013) (Koushanfar, 2012). Most of these schemes are based on the assumption that there is no direct access to the content of the key for legally activated chips (Rajendran et al., 2012b)(Baumgarten et al., 2010)(Chakraborty and Bhunia, 2008)(Rostami et al., 2013). In practice, however, powerful invasive and side-channel attacks can be applied for extracting the key information from read-proof memories (Rostami et al., 2013) (Joye, 2009) (Stanojlovic and Petkovic, 2010) (Rakers et al., 2001) (Tiri et al., 2002) (Moore et al., 2003). Ultimately, if a *common* key is used across all the fabricated chips, then such a key (that must be stored on every chip) becomes the most vulnerable part of the entire obfuscation mechanism. In other words, a single leaked key can compromise the entire security mechanism.

This threat motivates the approach of having a unique key for every chip, so that even if a key is leaked from some chip, it cannot be used to unlock other chips. A promising direction to address this problem is to use Physically Unclonable Functions (PUFs) (Herder et al., 2014). A PUF is a physical system, built based on the inherent process variations of chips at the manufacturing stage, which can be used as a unique signature/function for every chip. Nonetheless, engaging a PUF as a key against IC piracy is challenging, because a straightforward approach that uses the PUF only as a signature of the chip can be easily bypassed by the attacker/manufacturer (Alkabani and Koushanfar, 2007) (Wendt and Potkonjak, 2014).

The hardware obfuscation schemes in (Wendt and Potkonjak, 2014) and (Alkabani and Koushanfar, 2007) propose to modify the original design and engage a PUF as a part of the circuit's functionality. As each chip's PUF has a unique and unpredictable functionality, these schemes couple the PUF with a configurable module (constituting the key), which will be individually programmed for every chip by the designer during the post-fabrication activation process.

To program the configurable module of each and every chip, the designer needs to fully characterize the behavior of all the PUFs for all the chips, thus is constrained to use the PUFs with a limited search space, namely *weak* PUFs (Wendt and Potkonjak, 2014) (Alkabani and Koushanfar, 2007) (Herder et al., 2014). Unfortunately, an untrusted manufacturer is in a strong position of doing the same characterization for all the chips before handing them over to the designer for the activation process. In this case, when a leaked key is obtained for some activated chip, the attacker/manufacturer can easily recover the entire original design from: 1) the information of the leaked key, combined with 2) the pre-stored characterization of its associated PUF.

This chapter proposes to employ *strong* PUFs (with *huge* search space of truth-table) into the obfuscation framework, against such an attacker/manufacturer. While it is impossible for an attacker to fully characterize the strong PUFs, the main challenge becomes ensuring that the designer does not need to bear the burden of fully characterizing the strong PUFs to generate a unique key per-chip. This is achieved by employing an *Obfuscator* block into the design, which enables the designer to select an arbitrarily subset of the strong PUF to work, while guaranteeing that the architecture does not reveal to the attacker/manufacturer of the choices made by the designer. The chapter also discusses the security of the proposed scheme against several attacks, including the machine learning attacks (Ruhrmair et al., 2010) that are known to be powerful for characterizing many strong PUFs in a short time.

4.2 Preliminaries

Based on the number of CRPs that it can offer, each PUF belongs to one of the following two categories (Herder et al., 2014):

1. ***Weak** PUFs* offer a **limited** number of CRPs with respect to the number of their building components, such as SRAM-based PUF (Holcomb et al., 2009).
2. ***Strong** PUFs* offer a **huge** number of CRPs, i.e., exponential to the number of their building components, such as a delay-based Arbiter PUF (Devadas et al., 2008).

Since the entire truth-table of a weak PUF can be fully characterized via exhaustive evaluation, its CRPs must be kept secret from the potential attackers. Otherwise, once the entire truth-table is obtained,

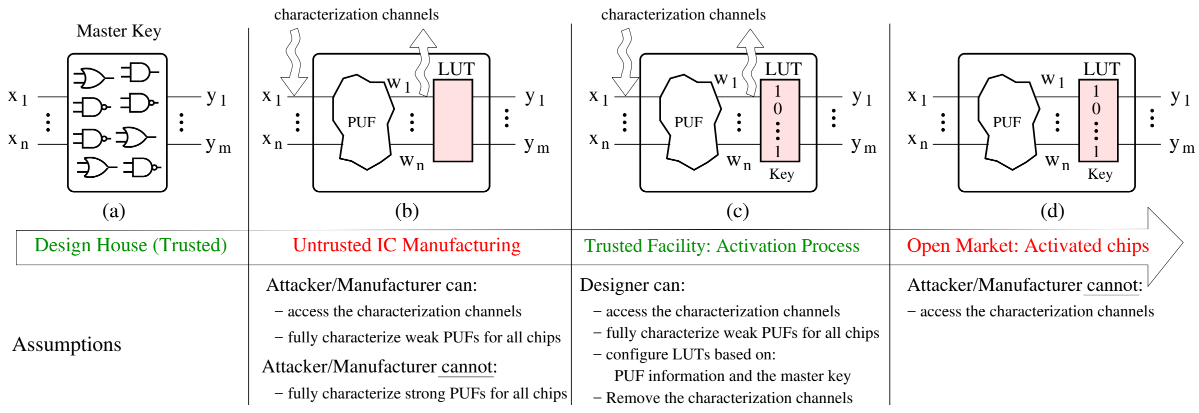

Master Key
characterization channels
characterization channels
x_1
x_n
y_1
y_m
PUF
LUT
w_1
w_n
Key
1
0
1
(a)
(b)
(c)
(d)
Design House (Trusted)
Untrusted IC Manufacturing
Trusted Facility: Activation Process
Open Market: Activated chips
Assumptions
Attacker/Manufacturer can:
– access the characterization channels
– fully characterize weak PUFs for all chips
Attacker/Manufacturer cannot:
– fully characterize strong PUFs for all chips
Designer can:
– access the characterization channels
– fully characterize weak PUFs for all chips
– configure LUTs based on: PUF information and the master key
– Remove the characterization channels
Attacker/Manufacturer cannot:
– access the characterization channels

Figure 23: Flow of PUF-based hardware obfuscation: (a) A part of the design with *n* inputs and *m* outputs is selected to be withheld as the master key. (b) The chip model with a PUF and a LUT, coupled to replace the master key. The manufactured chips based on this model will not be functional until activated by the designer. (c) The designer configures the LUT for each chip based on the behavior of the PUF to match the master key. (d) Activated chips in the open market.

the PUF is no longer unpredictable or unclonable, and its behavior can be emulated in software. For a strong PUF with an exponentially large number of CRPs, however, it is impossible to derive its entire truth-table via an exhaustive evaluation. Therefore, the security of an "ideal" strong PUF does not rely on keeping its CRPs secret, but rather on its exponentially huge truth-table (CRP space).

4.2.1 Machine learning attacks against strong PUFs

In theory the knowledge of a limited number of CRPs for an ideal PUF should not reveal any information about the responses to other untested challenges. However, this is not true in practice. As a matter of fact, *machine learning* classification algorithms (Kotsiantis et al., 2007) (Nasrabadi, 2007)(Abbasi et al., 2016) (Sharifzadeh et al., 2017) (Morente-Molinera et al., 2017) can be used to fully characterized many existing strong PUFs (Ruhrmair et al., 2010) (Becker, 2015), in which a precise model of the PUF is built in software after examining a limited number of CRPs.

The training size of such machine learning algorithms (which translates into the attack complexity) depends on the type of the PUF and the parameters of the PUF (Ruhrmair et al., 2010). For instance, the Arbiter PUF (Devadas et al., 2008) can be attacked in a linear time with respect to the number of its components. However, the training size to attack a different variant of the Arbiter PUF that uses nonlinearity in its architecture, known as the XOR Arbiter PUF (Zhou et al., 2017), grows exponentially with respect to the number of its components. As a result, while an Arbiter PUF with 128 sages (components) can be modeled in 2.10 seconds, modeling an XOR Arbiter PUF (with almost 5x hardware overhead) requires more than 16 hours (Ruhrmair et al., 2010). Another variant of the Arbiter PUF, known as the Lightweight PUF (Majzoobi et al., 2008), with a similar hardware cost, requires 267 days to be fully characterized (Ruhrmair et al., 2010).

Therefore, although machine learning attacks can model a wide variety of strong PUFs in a reasonable time with high accuracy, there exist promising secure PUFs against such attacks (Zhou et al., 2017) (Majzoobi et al., 2008). Thus, the underlying assumption of this chapter is that it is prohibitively expensive to characterize the behavior of **all** the strong PUFs for all the fabricated chips, where a such secure strong PUFs are employed.

4.2.2 Weak PUF-based hardware obfuscation

The work in (Wendt and Potkonjak, 2014) engages a PUF to achieve a unique key per chip. The basic flow is depicted in Fig. 23. It works by replacing a part of the circuit's functionality (the master key), shown in Fig. 23(a), with a PUF and a Look-Up Table (LUT), shown in Fig. 23(b). The LUT for each chip (serving as its key) will be uniquely configured by the designer in a post-fabrication activation process, shown in Fig. 23(c), so that the combination of the PUF and the LUT is functionally equivalent to the same common master key (withheld function) for all the chips. This approach guarantees that in case a key (the content of the LUT) is leaked from some chip, it cannot be simply inserted into other chips to activate them, because each LUT content is PUF-specific.

To configure the content of the LUT for each chip based on its unique PUF to achieve the functionality of the master key, the designer needs to obtain the PUF's CRPs. This is done via the peripheral characterization channels, shown in Fig. 23(c), during the activation process (Wendt and Potkonjak, 2014). Such channels provide a direct mechanism for the designer to apply the challenges and observe the responses. Since the inputs of the master key can potentially take any values from the internal signals in the circuit, the designer will need the knowledge of the *entire* CRP space (truth-table) of each PUF. Consequently, the designer is limited to the choice of a **weak** PUF, because the entire CRP space of a strong PUF is impractical to obtain.

4.3 Motivation

Based on the depicted flow in Fig. 23, in order to recover the master key, the attacker needs: 1) the information of a leaked key from some activated chip, and 2) the entire truth-table of the PUF for the same chip. It is usually assumed that the designer can remove all the PUF characterization channels from the chips after the activation process, so that there would be no physical channels left to characterize the PUFs for the legally activated chips in the open market, as shown in Fig. 23(d) (Wendt and Potkonjak, 2014) (Helfmeier et al., 2013). This can effectively prevent an attacker from obtaining the PUF truth-table for legally activated chips.

However, we argue that a malicious manufacturer can nonetheless gain access to the PUF at the fabrication stage (before sending the chips to the designer). At this stage, shown in Fig. 23(b), the characterization channels are fully accessible. Therefore, the attacker can obtain and store the truth-tables of all such weak PUFs for all the chips with a reasonable cost. Upon obtaining a leaked key from some activated chip in the market, the attacker can look up the truth-table of the PUF for the leaked chip in its database. Then, the attacker can derive the master key by combining the fully characterized truth-table of the PUF, and the LUT (leaked key) of the chip.

This threat motivates the usage of **strong** PUFs: their huge CRP space makes it prohibitively expensive both in terms of time and storage for an attacker to obtain the entire truth-table of all the PUFs (for all the chips) via the characterization channels during the manufacturing stage.

However, a straightforward adaptation of a strong PUF in the architecture in Fig. 23 is not feasible, as it would entail the same prohibitive characterization cost on the designer's side. Furthermore, since the functionality of the PUF is coupled with that of the LUT, switching from a weak PUF to a strong one implies a very large master key, thus increasing the implementation cost of the LUT exponentially. This is because adding a single input to the withheld function would double the required LUT size on chip.

Table I compares the usage of weak PUFs vs. strong PUFs in the obfuscation framework in terms of implementation cost of the LUT, as well as the PUF characterization costs on designer's side and attacker's side. As it can be seen from the table, the main challenge of adopting either of the PUFs is

TABLE I: Using weak PUFs vs strong PUFs in an obfuscation framework: an ideal scenario is marked out

	Weak PUF	Strong PUF
LUT cost (size of master key)	***LOW**	HIGH
PUF characterization for designer	***EASY**	HARD
PUF characterization for attacker	EASY	***HARD**

that they would entail the same level of characterization difficulty for the designer and the attacker. To achieve a secure, yet low cost obfuscation scenario, the following goals must be accomplished:

1. Small LUT size (i.e., the hardware cost).
2. Easy PUF characterization for designer.
3. Hard PUF characterization for attacker.

4.4 A Strong PUF-based Hardware Obfuscation

The key idea of the proposed scheme is to allow the designer to use a small **subset** of the entire CRP space of a strong PUF at the activation stage (as if the designer is dealing with a weak PUF), so as to maintain a low LUT cost. Meanwhile, the attacker is forced to deal with the **entire** CRP space of strong PUFs. To create such a "characterization gap" between the designer and the attacker, the designer's

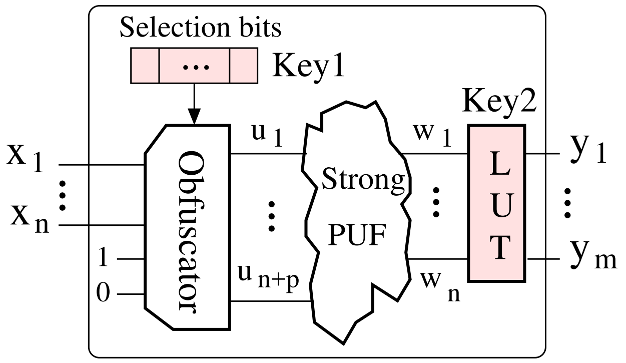


Figure 24: The architecture of the proposed scheme: by assigning Key1 for each chip, a permutation of the original wires appears at n inputs of the PUF, and the other p inputs of the PUF are set with arbitrarily specified values. Only a subset of the PUF with n inputs and n outputs will be used by the designer. The LUT (Key2) is programmed to compensate for the unique behavior of the PUF. The unique key per chip is $\langle$Key1 , Key2$\rangle$.

selections of the subset of the PUF for each chip must be hidden from the attacker at the manufacturing stage.

Such a secure subset selection is achieved by employing an *Obfuscator* coupled with the strong PUF in the obfuscation framework. Fig. 24 depicts the main architecture of the proposed scheme. Suppose that the master key (withheld function) has n inputs and m outputs (the same as in the case of a weak PUF), such that 2^n is a reasonable space. To maintain the same amount of hardware for implementing the LUT, the strong PUF must have the same n outputs as the weak PUF. The number of inputs of the strong PUF, however, is set to $n+p$, in such a way that $p >> n$, so that the entire truth-table of the PUF (with 2^{n+p} CRPs) can be increased exponentially by increasing the value of p.

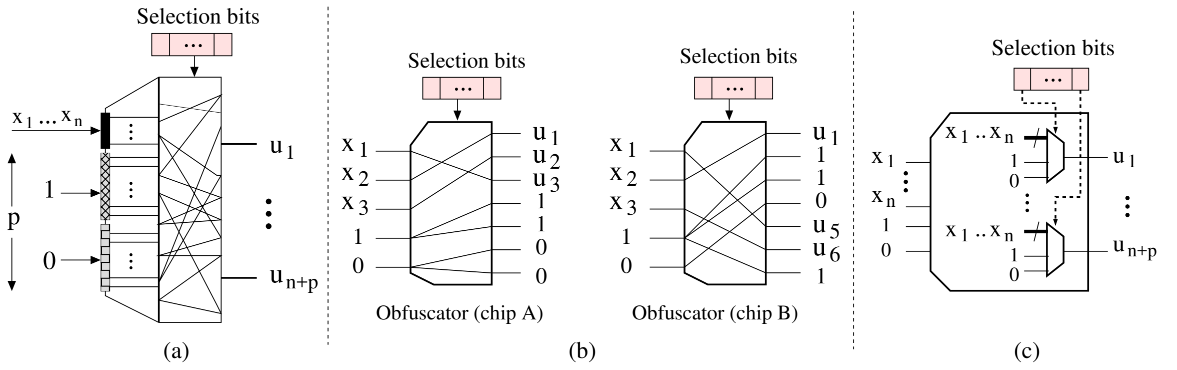


Figure 25: (a) Obfuscator: By specifying the Selection bits, a permutation of n original wires, as well as p fixed values (0 or 1) will appear at the output of the block; (b) Examples of an Obfuscator block (with $n = 3$ and $p = 4$) to allow the designer to select any subset of the CRP space during the activation process without revealing it to the attacker at the manufacturing stage; (c) A possible MUX-based implementation of the Obfuscator

The functionality of the Obfuscator, shown in Fig 25(a), enables the designer to use any subset (with n inputs) of the entire CRP space of the strong PUF for each chip by fixing the values of p inputs during the activation process, and collecting the truth-table information of the PUF with the remaining n inputs, a total of 2^n CRPs (the same as the case of employing a weak PUF). The configuration of the Obfuscator is determined by: 1) scrambling the n original wires, signals $\{x_1, x_2, ..., x_n\}$, such that a permutation of them appears at the inputs of the PUF; and 2) sending arbitrarily specified values to the other p inputs of the PUF. The specific configuration of each chip's Obfuscator is set by the *Selection bits* block, during the activation process. Fig. 25(b) shows two small examples of an Obfuscator with different configurations.

Overall, the Obfuscator ensures that: 1) the designer is able to select an arbitrarily CRP subset (with n inputs) of the strong PUF during the activation process for each chip; and 2) the architecture does not reveal to the attacker during the manufacturing stage which subset of the PUF will be selected later on. As a result, the unique key per chip consists of two elements: 1) the subset selection of the strong PUF, i.e., the content of the Selection bits (Key1); and 2) the content of the LUT (Key2) configured based on the functionality of the selected subset of the PUF.

At the manufacturing stage, the manufacturer cannot identify the subset of the strong PUF that will be used by the designer for each chip. Therefore, to get a full database of the CRP space of all PUFs, the attacker/manufacturer has to examine and store all 2^{n+p} possible CRPs for every PUF, which is prohibitively costly.

At the activation stage, by specifying the content of Selection bits (Key1), n inputs of the PUF will be connected to a permutation of the original wires of the master key, $\{x_1, x_2, ..., x_n\}$, and the rest will be fully specified for every chip. With the p fixed inputs, a total of 2^n CRPs remains for each PUF. The designer can then examine all 2^n CRPs (as if it was a weak PUF) using the characterization channels to determine the content of the LUT (Key2) for every chip, so that the combination of the selected subset of the strong PUF and LUT is functionality equivalent with the withheld function.

It must be noted that while the designer needs to check 2^n different CRPs to derive the unique key of every chip at the activation stage, the attacker is forced to examine and store all 2^{n+p} possible CRPs ($p >> n$) for every PUF at the manufacturing stage. At the end of the activation process, all the characterization channels to the PUF are removed to prevent any further characterization of the PUFs for activated chips. This can be achieved by laser burning access wires or burning supporting fuses (Helfmeier et al., 2013) (Wendt and Potkonjak, 2014).

Fig. 25(c) shows a general implementation of the Obfuscator block. This block can be implemented with $n+p$ Multiplexers (MUXes). Each MUX selects a few inputs (less or equal to n) from the original inputs of the master key and the fixed values (logic "1" or "0") to output to an input bit of the PUF.

Furthermore, this architecture can employ a relatively large number of "dummy" fan-outs, say q, from the original gates of the design, which will be connected to the inputs of the Obfuscator block (along with the n original signals) (Khaleghi et al., 2015). The propagation of these dummy fan-outs will be blocked by the Obfuscator. Therefore, if an attacker wants to model the entire scheme with a virtual LUT, the q dummy fan-outs increases the search space of the attacker exponentially (with $O(2^{n+q})$ complexity) at a linear hardware overhead of the fan-outs (Khaleghi et al., 2015).

4.5 Security Analysis

4.5.1 Attack model

The proposed scheme and discussion of this chapter is based on the assumptions that the attacker has: 1) the complete gate-level net-list; 2) full knowledge of the security scheme; 3) access to the activated chips from the open market; 4) access to the content of on-chip LUT (via side-channel attacks, etc); 5) access to the PUF characterization channels at the manufacturing stage. On the other hand, this chapter assumes that the attacker ***cannot***: 1) access the PUF characterization channels of legally activated chips; 2) fully characterize strong PUFs for **all** the chips at the manufacturing stage.

4.5.2 Attack analysis upon a leaked key

In the proposed scheme, the unique key per chip consists of two parts: Key1, the content of the Selection bits, and Key2, the content of the LUT. Consider the worst case that the attacker has achieved a copy of the entire key, $\langle$Key1 , Key2$\rangle$, for a particular chip. It is obvious that this key cannot be used directly to activate other chips. In order to recover the master key, the attacker has to examine the CRP space of the subset of the PUF that is used by the designer for the leaked chip. Next, we will discuss various possible attacks under such a scenario:

4.5.2.1 PUF characterization of chips in open market

After obtaining a leaked key $\langle$Key1 , Key2$\rangle$ from some chip, the attacker would be able to identify the subset of the PUF that is used by the designer for that chip (from analyzing Key1). However, as all the characterization channels of the PUFs have been removed at the end of the activation process, the PUF can be no longer characterized.

4.5.2.2 PUF characterization of chips at manufacturing stage

At the manufacturing stage, the characterization channels are available to the attacker. However, since the chips are not activated yet, the attacker cannot have the “leaked key” to help indicate which subset of the PUF will be used. The huge CRP space of the strong PUFs ensures that it is prohibitively expensive to exhaustively examine all the CRPs even for a single PUF. As it was discussed in section 4.2.1, it is also prohibitively expensive for a manufacturer to perform machine learning attacks on **all** the fabricated chips, where a secure strong PUF is employed.

4.5.2.3 SAT-based attacks

Without a direct way to obtain the CRP space of the PUFs, the attacker can model the entire security block (Obfuscator, PUF and the LUT) with a *virtual* LUT, and then try to find the content of such LUT by applying carefully designed primary inputs to a working chip and analyzing the values of the primary outputs (Subramanyan et al., 2015). The key idea to overcome such *SAT-based* attacks is to carefully select the withheld function at the design stage, so that the outputs of the LUTs become strongly correlated (Rajendran et al., 2012b) (Baumgarten et al., 2010) (Khaleghi et al., 2015). The proposed scheme in this work can work with many SAT-based prevention schemes to achieve a stronger framework. Furthermore, the designer can increase the number of q dummy fan-outs fed to the Obfuscator (as it was explained in section 4.4), so as to increase the cost of SAT-based attacks by enlarging the size of the virtual LUT exponentially at a linear cost.

4.5.3 Attack complexity

The attack complexity for PUF characterization (in terms of time and storage) at the manufacturing stage is $O(2^{n+p})$ multiplied by the number of fabricated chips. If the attacker decides to bypass the entire scheme, the attack model has to tackle an enlarged virtual LUT of $n+q$ address bits. This effectively boost the total search space to be $O(m \times 2^{n+q})$, which is 2^q times larger than the complexity of the hardware cost imposed on the designer's side. Therefore, the attacking costs both at the manufacturing stage and after obtaining a leaked key can be controlled by the designer through tuning the parameters p and q, respectively.

4.6 Cost and Implementation Discussion

4.6.1 Hardware cost complexity

The hardware cost complexity in the general case of a withheld function with n inputs, m outputs along with the PUF and Obfuscator includes: the implementation cost of the LUT (Key2), of $O(m \times 2^n)$ complexity, plus the cost for the Obfuscator block, of $O((n+p) \times n)$ complexity (including $n+p$ MUXes and the interconnect network). The cost of the Selection bits block (Key1) would be of $O((n+p) \times log(n))$. The PUF architecture would have the cost complexity of $O(n+p)$. Overall, as it is shown in Fig. 24, the LUT cost of implementing the withheld function will remain unchanged (compared to the weak PUF implementation).

4.6.2 PUF reliability

One of the important factors that need to be considered for any PUF-based scheme is the noisiness of PUF devices. It is usually suggested that using an Error Correcting Code (ECC) (Yu and Devadas, 2010) can compensate for the noisiness of the PUF. In this particular scheme, the existence of many redundant CRPs in a strong PUF can be used to reduce (not eliminate) the cost of ECC: it has been shown that some of the challenge bits of PUFs may show an unstable behavior for many cases. Such an issue can be handled by avoiding the use of the unstable challenge bits of the PUF, through employing an alternate, yet similar architecture, in which the LUT *precedes* the PUF. This way, the unstable challenge bits of the PUF can be avoided by configuring the content of the LUT (Key2) accordingly.

4.6.3 Performance overhead

From the timing perspective, replacing a part of the design with a PUF coupled with a LUT can significantly increase the delay of the overall design, if the withheld function lies in a critical path. In order to avoid such timing violations, one must carefully select the withheld function to avoid the critical paths.

4.7 Evaluation

This section evaluates the effectiveness of the proposed scheme by comparing the hardware overhead for the designer and the cost to attack. The experiments are run on a number of ITC'99 benchmarks (Corno et al., 2000). The size of the LUT (Key2) is fixed to be 32-bit ($n = 5$) for all the experiments. This is to ensure that the time complexity for the designer to derive the key is reasonable. The strong PUF is built based on the proposed XOR Arbiter in (Zhou et al., 2017). Fig. 26 shows the attacking cost against hardware overhead for three benchmarks [1]. Two types of attacks are considered: 1) the attack at manufacturing stage to obtain the entire-truth table information of a PUF, and 2) the attack to crack the security scheme after the key leaks out from some chip. Different curves on each plot are derived by increasing parameter p; thus, all points on each curve have the same attacking cost at the manufacturing stage (equal to 2^{n+p} CRPs). The cost to attack after a leaked key is obtained from one chip is depicted on the y-axis in terms of the size of the virtual LUT (determined jointly by parameters q and n) that must be cracked by the attacker.

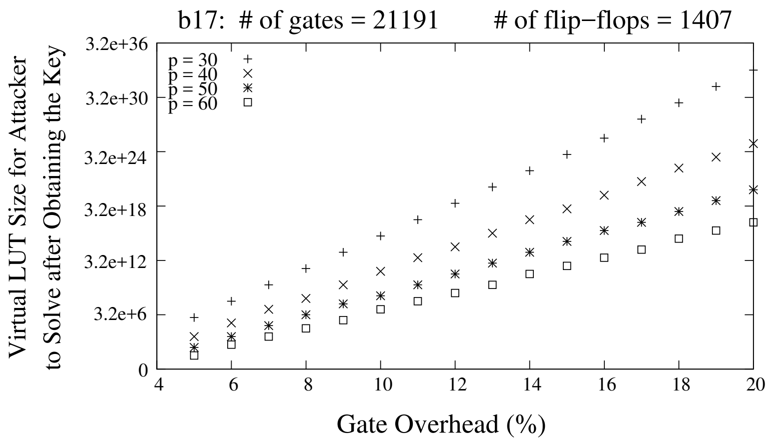
The logarithmic scale of the y-axis in Fig. 26 indicates that the cost to attack at the manufacturing stage and upon obtaining a leaked key can be increased exponentially at a linear hardware overhead. Also, it can be seen that given a fixed hardware overhead, the designer can increase the difficulty of one

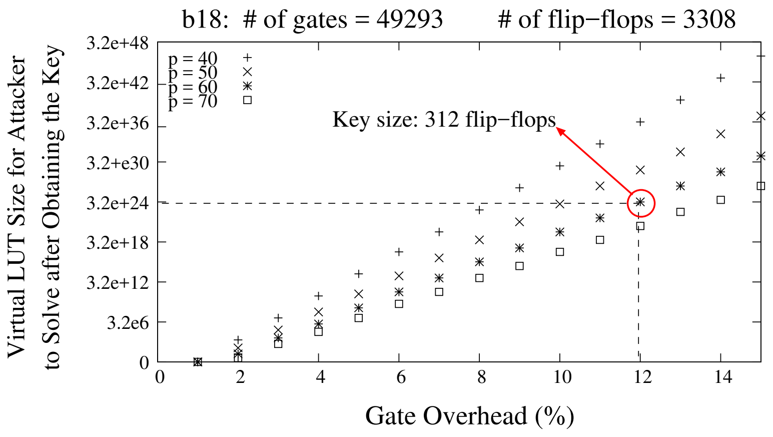
[1]The reported hardware overheads do not take into account the implementation costs of Error Correcting Codes (ECC) to emphasize on the overhead of the proposed scheme; however, the ECC overhead must be added to the overall overhead of the system.

of the attacks at the cost of decreasing the difficulty of the other attack. However, the designer is in full control of tuning the parameters of the scheme (*p*, *q*, and *n*) at the design stage, so that both attacks become prohibitively expensive. Furthermore, Fig. 26 shows the scalability of the proposed scheme: A larger circuit (b19) can achieve the same level of security as a smaller circuit (b18) at a lower hardware overhead (6% for b19 compared to 12% for b18). Here, the total size of the key contains 312 flip-flops for both circuits, which is reasonable compared to the total number of flip-flops in each design.

4.8 Conclusion

In this chapter, we argued that a weak PUF-based obfuscation is subject to attacks from a manufacturer that can: 1) pre-collect the entire truth-table of the PUFs for all the chips at the fabrication stage with a reasonable cost; and 2) using the information of a leaked key from an activated chip, to recover the master key. The proposed scheme engages a strong PUF in the obfuscation scheme, and therefore relies on the prohibitively expensive cost for an attacker to characterize and store all the strong PUFs for all the fabricated chips. While the cost to attack grows exponentially for the attacker, the hardware cost grows linearly for the designer to adopt such a scheme, therefore making it a viable and scalable solution against IC piracy attacks.





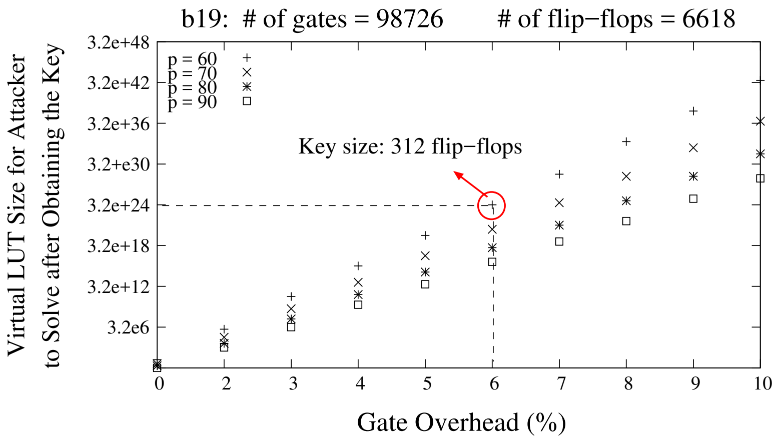


Figure 26: The cost to attack vs. hardware overhead of the proposed scheme for different circuits.

CHAPTER 5

CONCLUSION

Parts of this chapter have been presented in (Khaleghi et al., 2015), (Khaleghi et al., 2016), (Khaleghi and Rao, 2018). Copyright © 2015, 2016, 2018, IEEE.

IC piracy is a significant security threat, where malicious manufacturers can produce unauthorized extra chips and/or steal the information of a design through reverse engineering attempts.

As a countermeasure, hardware obfuscation schemes usually withhold a part of the design (which thereafter constitutes the "key") by replacing it with configurable modules, 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. Enforcing the configurable module to be filled in with the withheld key information enables a post-manufacturing activation of each authenticate chip allows the designer to control the number of functioning chips in market.

However, most existing obfuscation approaches are ad-hoc based, and are facing two major challenges: 1) *algorithmic attacks* applied on the obfuscated design, that could potentially crack the keys efficiently, and 2) *physical attacks* applied on the unlocked chips, aiming at reading out the keys directly from the on-chip memory cells.

To address these two challenges, the proposed work aims at simultaneously *expanding the control of a designer*, and *restraining the control of an attacker*, throughout the IC design and fabrication process. This is approached in the following ways: 1) by systematic entangling the various obfuscation primitives in a hierarchical manner, a strong defense mechanism can be built to ensure that the cost of algorithmic attacks (in terms of computational complexity) can be raised exponentially, while the designer's cost (in terms of hardware overhead on chip) only increases linearly; 2) against the worst-case scenario of combined algorithmic and physical attacks, a preventive architecture is proposed to deliver a unique key per chip, via the engagement of Physically Unclonable Functions (PUFs) into the obfuscation paradigm. The architecture and integration with the obfuscation primitives allows a PUF to be harnessed to equip each chip with a unique key, with controls by the designer while out of the controls of a potential attacker. This will ensure that even a completely leaked key cannot be used for piracy purposes. Overall, the deliverable security (in terms of attack costs) and defense costs (as overhead on the designer's side) can be quantitatively modeled, analyzed, and proved, in an asymptotic manner, making it suitable for the scalability of IC design, serving a strong basis for a "design against piracy" framework.

Appendices

Appendix A

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis. The list of the articles include IEEE Asia and South Pacific Design Automation Conference (ASP-DAC) 2015 (Khaleghi et al., 2015), IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH) 2016 (Khaleghi et al., 2016), and IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2018 (Khaleghi and Rao, 2018) conference papers. The permissions follow in the same order.

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title:	IC Piracy prevention via Design Withholding and Entanglement
Conference Proceedings:	The 20th Asia and South Pacific Design Automation Conference
Author:	Soroush Khaleghi
Publisher:	IEEE
Date:	Jan. 2015

Copyright © 18, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.

Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Create Account | Help

IEEE Requesting permission to reuse content from an IEEE publication

Title:	An STT-MRAM based strong PUF
Conference Proceedings:	2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)
Author:	Soroush Khaleghi
Publisher:	IEEE
Date:	July 2016

Copyright © 2016, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.
Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink® Home Create Account Help

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Hardware Obfuscation Using Strong PUFs
Conference Proceedings:	2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)
Author:	Soroush Khaleghi
Publisher:	IEEE
Date:	Jul 2018

Copyright © 2018, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials.

Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CITED LITERATURE

[Abbasi et al. , 2016] Abbasi, B., Noohi, E., Parastegari, S., and Žefran, M.: Grasp taxonomy based on force distribution. In Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on, pages 1098–1103. IEEE, 2016.

[Abramovici et al. , 1990] Abramovici, M., Breuer, M. A., and Friedman, A. D.: Digital Systems Testing and Testable Design. IEEE Press, 1990.

[Alkabani and Koushanfar, 2007] Alkabani, Y. and Koushanfar, F.: Active hardware metering for intellectual property protection and security. USENIX Security, pages 291–306, 2007.

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

[Baumgarten et al. , 2010] Baumgarten, A., Tyagi, A., and Zambreno, J.: Preventing ic piracy using reconfigurable logic barriers. IEEE Design and Test Computers, 27:66–75, 2010.

[Becker, 2015] Becker, G. T.: The gap between promise and reality: on the insecurity of xor arbiter pufs. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 535–555. Springer, 2015.

[Caldwell et al. , 2004] Caldwell, A., Choi, H.-J., Kahng, A., Mantik, S., Potkonjak, M., Qu, G., , and Wong, J.: Effective iterative techniques for fingerprinting design ip. 23(2):208–215, 2004.

[Chakraborty and Bhunia, 2008] Chakraborty, R. S. and Bhunia, S.: Hardware protection and authentication through netlist level obfuscation. IEEE/ACM Conference on Computer-Aided Design, pages 674–677, 2008.

[Chakraborty and Bhunia, 2009] Chakraborty, R. S. and Bhunia, S.: HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 28(10):14931502, 2009.

[Chipworks, 2012] Chipworks: Intels 22-nm Tri-gate Transistors Exposed, 2012.

[Corno et al. , 2000] Corno, F., Reorda, M., and Squillero, G.: Rt-level itc'99 benchmarks and first atpg results. Design Test of Computers, IEEE, 17(3):44–53, Jul 2000.

[DARPA, 2012] DARPA: Integrity and Reliability of Integrated Circuits (IRIS), 2012.

[Devadas et al. , 2008] Devadas, S., Suh, E., Paral, S., Sowell, R., Ziola, T., and Khandelwal, V.: Design and implementation of puf-based unclonable rfid ics for anti-counterfeiting and security applications. IEEE International Conference on RFID, pages 58–64, April 2008.

[Erb et al. , 2013] Erb, D., Kochte, M., Sauer, M., H.Wunderlich, and Becker, B.: Accurate multi-cycle atpg in presence of x-values. Asian Test Symposium (ATS), pages 250–245, 2013.

[Fleet and Dransfield, 1998] Fleet, W. M. V. and Dransfield, M. R.: Method of recovering a gate-level netlist from a transistor-level. US Patent no. 6190433, 1998.

[Fujiwara and Toida, 1982] Fujiwara, H. and Toida, S.: The complexity of fault detection problems for combinational logic circuits. IEEE Transactions on Computers, pages 555–560, 1982.

[Helfmeier et al. , 2013] Helfmeier, C., Nedospasov, D., Tarnovsky, C., Krissler, J., Boit, C., and Seifert, J.: Breaking and entering through the silicon. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pages 733–744. ACM, 2013.

[Herder et al. , 2014] Herder, C., Yu, M., Koushanfar, F., and Devadas, S.: Physical unclonable functions and applications: A tutorial. proceedings of the IEEE, 102:1126–1141, May 2014.

[Holcomb et al. , 2009] Holcomb, D., Burleson, W., and Fu, K.: Power-up sram state as an identifying fingerprint and source of true random numbers. 58(9):1198–1210, 2009.

[Iyengar et al. , 2016] Iyengar, A., Ghosh, S., Rathi, N., and Naeimi, H.: Side channel attacks on sttram and low-overhead countermeasures. In Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2016 IEEE International Symposium on, pages 141–146. IEEE, 2016.

[Joye, 2009] Joye, M.: Basics of side-channel analysis. Cryptographic Engineering, pages 365–380, 2009.

[Kahng et al. , 1998a] Kahng, A., Lach, J., Smith, W. M., Mantik, S., Makrov, I., Potkonjak, M., Tucker, P., Wang, H., and Wolfe, G.: Watermarking techniques for intellectual property protection. IEEE/ACM Design Automation Conference, pages 776–781, 1998.

[Kahng et al. , 1998b] Kahng, A., Mantik, S., Makrov, I., Potkonjak, M., Tucker, P., Wang, H., and Wolfe, G.: Robust ip watermarking methodologies for physical design. IEEE/ACM Design Automation Conference, pages 782–787, 1998.

[Khaleghi et al. , 2016] Khaleghi, S., Vinella, P., Banerjee, S., and Rao, W.: An stt-mram based strong puf. In Nanoscale Architectures (NANOARCH), 2016 IEEE/ACM International Symposium on, pages 129–134. IEEE, 2016.

[Khaleghi et al. , 2015] Khaleghi, S., Zhao, K. D., and Rao, W.: Ic piracy prevention via design withholding and entanglement. IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), pages 821–826, 2015.

[Khaleghi and Rao, 2018] Khaleghi, S. and Rao, W.: Hardware obfuscation using strong pufs. In 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pages 321–326. IEEE, 2018.

[Kotsiantis et al. , 2007] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P.: Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160:3–24, 2007.

[Koushanfar, 2012] Koushanfar, F.: Hardware metering: A survey. In Introduction to Hardware Security and Trust, pages 103–122. Springer, 2012.

[Koushanfar et al. , 2005] Koushanfar, F., Hong, I., and Potkonjak, M.: Behavioral synthesis techniques for intellectual property protection. 10(3):523–545, 2005.

[Lach et al. , 1998] Lach, J., Mangione-Smith, W., and Potkonjak, M.: FPGA fingerprinting techniques for protecting intellectual property. IEEE Custom Integrated Circuits Conference, pages 299–302, 1998.

[Lee and Ha, 1991] Lee, H. and Ha, D.: An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation. Proc. of IEEE International Test Conference, pages 946–955, 1991.

[Majzoobi et al. , 2008] Majzoobi, M., Koushanfar, F., and Potkonjak, M.: Lightweight secure pufs. In IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 670–673. IEEE, 2008.

[Moore et al. , 2003] Moore, S., Anderson, R., Mullins, R., Taylor, G., and Fournier, J. J.: Balanced self-checking asynchronous logic for smart card applications. Microprocessors and Microsystems, 9(27):421430, 2003.

[Morente-Molinera et al. , 2017] Morente-Molinera, J. A., Mezei, J., Carlsson, C., and Herrera-Viedma, E.: Improving supervised learning classification methods using multi-granular linguistic modelling and fuzzy entropy. IEEE transactions on fuzzy systems, 25(5):1078–1089, 2017.

[Nasrabadi, 2007] Nasrabadi, N. M.: Pattern recognition and machine learning. Journal of electronic imaging, 16(4):049901, 2007.

[Office, 2011] Office, U. G. P.: The committees investigation into counterfeit electronic parts in the department of defense supply chain, 2011.

[Rajendran et al. , 2012a] Rajendran, J., Pino, Y., Sinanoglu, O., and Karri, R.: Logic encryption: A fault analysis perspective. Design, Automation and Test in Europe, page 953958, 2012.

[Rajendran et al. , 2012b] Rajendran, J., Pino, Y., Sinanoglu, O., and Karri, R.: Security analysis of logic obfuscation. IEEE/ACM Design Automation Conference, pages 83–89, 2012.

[Rakers et al. , 2001] Rakers, P., Connell, L., Collins, T., and Russell, D.: Secure Contactless Smartcard ASIC with DPA protection. Journal of Solid-State Circuits, page 559565, 2001.

[Rivest et al. , 1978] Rivest, R., Shamir, A., and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[Rostami et al. , 2013] Rostami, M., Koushanfar, F., Rajendran, J., and Karri, R.: Hardware security: Threat models and metrics. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 819–823, 2013.

[Roy and Koushanfar, 2008] Roy, J. and Koushanfar, F.: Epic: Ending piracy of integrated circuits. Design, Automation and Test in Europe, pages 1069–1074, 2008.

[Ruhrmair et al. , 1011] Ruhrmair, U., Devadas, S., and Koushanfar, F.: Security based on Physical Unclonability and Disorder. Book Chapter in Introduction to Hardware Security and Trust, 1011.

[Ruhrmair et al. , 2010] Ruhrmair, U., Sehnke, F., Solter, J., Dror, G., Devadas, S., and Schmidhuber, J.: Modeling attacks on physical unclonable functions. Proceedings of the 17th ACM conference on Computer and communications security, pages 237–249, 2010.

[Samyde et al. , 2002] Samyde, D., Skorobogatov, S., Anderson, R., and Quisquater, J.-J.: On a new way to read data from memory. In Security in Storage Workshop, 2002. Proceedings. First International IEEE, pages 65–69, Dec 2002.

[Sharifzadeh et al. , 2017] Sharifzadeh, M., Agarwal, C., Aloraini, M., and Schonfeld, D.: Convolutional neural network steganalysis's application to steganography. In Visual Communications and Image Processing (VCIP), 2017 IEEE, pages 1–4. IEEE, 2017.

[Skorobogatov, 2009] Skorobogatov, S.: Local heating attacks on flash memory devices. In Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on, pages 1–6, July 2009.

[Skorobogatov, 2010] Skorobogatov, S.: Optical fault masking attacks. In Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on, pages 23–29, Aug 2010.

[Stanojlovic and Petkovic, 2010] Stanojlovic, M. and Petkovic, P.: Strategies against side-channel attack. Small Systems Simulation Symp, pages 86–89, 2010.

[Subramanyan et al. , 2015] Subramanyan, P., Ray, S., and Malik, S.: Evaluating the security of logic encryption algorithms. In Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on, pages 137–143. IEEE, 2015.

[Sush and Devadas, 2007] Sush, G. and Devadas, S.: Physical unclonable functions for device authentication and secret key generation. IEEE/ACM Design Automation Conference, pages 9–14, 2007.

[Tiri et al. , 2002] Tiri, K., Akmal, M., and Verbauwhede, I.: A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. European Solid-State Circuits Conference, page 403406, 2002.

[Torrance and James, 2011] Torrance, R. and James, D.: The state-of-the-art semiconductor reverse engineering. IEEE/ACM Design Automation Conference, pages 333–338, 2011.

[Wendt and Potkonjak, 2014] Wendt, J. and Potkonjak, M.: Hardware obfuscation using puf-based logic. Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on, pages 270–271, November 2014.

[Wolf et al. , 2010] Wolf, S., Lu, J., Stan, M., Chen, E., and Trege, D.: The promise of nanomagnetics and spintronics for future logic and universal memory. proceedings of the IEEE, pages 2155 – 2168, 2010.

[Yu and Devadas, 2010] Yu, M. and Devadas, S.: Secure and robust error correction for physical unclonable functions. Design Test of Computers, IEEE, 27(1):48–65, Jan 2010.

[Zamanzadeh and Jahanian, 2013] Zamanzadeh, S. and Jahanian, A.: Automatic netlist scrambling methodology in asic design flow to hinder the reverse engineering. IFIP/IEEE Very Large Scale Integration, pages 52–54, 2013.

[Zhang et al. , 2014] Zhang, L., Fong, X., Chang, C.-H., Kong, Z., and Roy, K.: Highly reliable memory-based physical unclonable function using spin-transfer torque mram. IEEE International Symposium on Circuits and Systems (ISCAS), pages 2169 – 2172, 2014.

[Zhou et al. , 2017] Zhou, C., Parhi, K., and Kim, C.: Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements. IEEE/ACM Design Automation Conference (DAC), 2017.

VITA

Name Soroush Khaleghi

Education **Ph.D.**, Electrical and Computer Engineering
University of Illinois at Chicago, Chicago, Illinois, United States, 2018

M.Sc., Electrical and Computer Engineering
University of Illinois at Chicago, Chicago, Illinois, United States, 2017

B.Sc., Electrical Engineering
Sharif University of Technology, Tehran, Iran, 2012

Publications **S. Khaleghi**, W. Rao, "*Hardware Obfuscation Using Strong PUFs*", IEEE Computer Society Annual Symposium on VLSI (ISVLSI), July 2018.

S. Khaleghi, Y. Yin, R. Hadad, "*Improving Computational Thinking with Arduino Activities*", American Educational Research Association Annual Meeting (AERA), April 2018.

S. Khaleghi, P. Vinella, S. Banerjee, W. Rao, "*An STT-MRAM based Strong PUF*", International Symposium on Nanoscale Architectures (NANOARCH), 2016.

S. Khaleghi, K. D. Zhao, W. Rao, "*IC Piracy Prevention via Design Withholding and Entanglement*", Asia and South Pacific Design Automation Conference (ASP-DAC), Pages 821-826, 2015.

S. Khaleghi, W. Rao, "*Spare Sharing Network Enhancement for Scalable Systems*", *IEEE* International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Pages 249-254, 2013.

S. Khaleghi, W. Rao, "*Constructing Spare Sharing Network for Reliability Enhancement of Scalable Systems*", *IEEE* Workshop on Signal Processing Systems (SiPS), Pages 336-341, 2013.