KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Differential Scan-Based Side-Channel Attacks and Countermeasures

Secure Design-for-Testability (DfT) for Cryptographic Circuits

Amitabh Das

Dissertation presented in partial
fulfillment of the requirements for the
degree of Doctor in Engineering

October 2013

Differential Scan-Based Side-Channel Attacks and Countermeasures

Secure Design-for-Testability (DfT) for Cryptographic Circuits

Amitabh DAS

Supervisory Committee:
Prof. dr. ir. Pierre Verbaeten, chair
Prof. dr. ir. Ingrid Verbauwhede, supervisor
Prof. dr. ir. Wim Dehaene
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Nele Mentens
Prof. dr. ir. Bruno Rouzeyre
(LIRMM, University of Montpellier (UM2), Montpellier, France)
Ir. Erik Jan Marinissen, PDEng
(Principal Scientist, IMEC, Leuven, Belgium)

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

October 2013

Kasteelpark Arenberg 10, box 2452, B-3001 Heverlee (Belgium)

D/2013/7515/121
ISBN 978-94-6018-736-0

Preface

This PhD thesis is about secure testing of cryptographic hardware. Security and testability are two conflicting goals that are difficult to achieve simultaneously. The purpose of this dissertation and my PhD research has been mainly to address this conflict, while at the same time creating feasible low-cost solutions that can be implemented in real applications.

I would like to express my sincere gratitude to Prof. Ingrid Verbauwhede for giving me the opportunity to pursue research at the COSIC group of KU Leuven. Her guidance and encouragement in these four years have been instrumental in architecting my PhD. I would also like to thank my assessors, Prof. Wim Dehaene and Prof. Bart Preneel for giving their insights and asking stimulating questions during the course of my PhD research that has helped in further improving this work. I would also like to acknowledge my other jury members, Erik Jan Marinissen at IMEC, Prof. Nele Mentens, and Prof. Bruno Rouzeyre from LIRMM, Montpellier, France, for their improvement suggestions and support. My collaboration with Prof. Rouzeyre's research group at LIRMM and joint work with my co-author, Jean Da Rolt, helped a lot in developing my PhD research. I also acknowledge my other co-authors, especially Baris Ege, Santosh Ghosh and Stefaan Seys, for their efforts in our joint publications.

I am thankful to Pela, our COSIC group secretary, for helping in the administrative process at KU Leuven, and in my personal problems during my PhD years. I am also grateful for the support of my colleagues at COSIC who have been kind to guide and help me in my research. This PhD would not have been possible without the continuous encouragement of my parents, my elder brother, and my wife, who have played a major role in building my career. I am really indebted to them for inspiring and motivating me all along my PhD.

Amitabh Das
Leuven-Heverlee, Belgium, September 2013

Abstract

Cryptographic circuits are vulnerable to various side-channel attacks that target their hardware implementations to extract secret information stored inside them. One such side-channel is the scan chain based Design-for-Test (DfT) infrastructure employed for thorough and faster testing of VLSI circuits. Removing the connectivity of scan chains after manufacturing test prevents such attacks, but also makes in-field test and updates of the circuits impossible. In some applications, such as set-top box decoders, the firmware updates happen through the JTAG port internally connected to the scan chains. Hence, scan chains must be left intact and at the same time protected from these attacks. Moreover, the cost in terms of area and test time overhead must be kept to a minimum to make it feasible to incorporate the security mechanism on a reasonably priced commercial product. This work first investigates the scan attack vulnerability of symmetric-key and public-key hardware implementations, and then presents suitable countermeasures to address the aforementioned trade-off between testability, security and test cost.

The thesis first presents scan attacks on hardware implementations of the symmetric-key block cipher AES and the public-key ciphers RSA and ECC in the presence of advanced DfT structures such as test compression and X-handling schemes. In addition, state-of-the-art power analysis side-channel and fault attack countermeasures are analyzed to evaluate whether they are suitable in warding off scan attacks. The thesis also investigates the practical security provided by various scan attack countermeasures (such as partial scan and scan chain scrambling) that are proposed in the literature. At the algorithmic level, blinding and randomization based schemes that protect against Differential Power Analysis (DPA) attacks are shown to be secure against scan attacks, whereas countermeasures against Simple Power Analysis (SPA) and Fault Attacks are found to be ineffective against scan attacks. At the Register Transfer level (RTL), Multiple Input Signature Register (MISR)-based time compaction schemes are found be inherently secure against scan attacks, provided only the final MISR signature is observable and not the intermediate states. New countermeasures are also proposed at the system level in the form of a secure JTAG architecture based on an ECC-based Schnorr Protocol and at the gate level in the form of a noise injector integrated with the test compression schemes.

Another major contribution of the thesis is secure Cryptographic SoC Testing. As part of our research work, scan attack resistant Secure Test Wrappers (STWs) have been designed that integrate a challenge-response based secure entity authentication protocol with the IEEE 1500 standard Test Wrapper. Two variants of STWs are proposed; one based on a lightweight block cipher KATAN and the other using Physically Unclonable Functions (PUFs). Another work performed in this direction is integrating efficient multiplier-based pseudo-random Logic Built-In Self-Test (LBIST) solutions with STWs. This helps in providing a flexible self-testing option for the cryptographic SoC and maintaining a high level of testability and security, while simultaneously reducing the test overhead.

Samenvatting

Cryptografische schakelingen zijn vatbaar voor verscheidene nevenkanaalsaanvallen met als doel de extractie van geheime sleutels die opgeslagen zijn in de hardware implementatie. Eén van deze nevenkanalen is de “Design-for-Test (DfT)” infrastructuur die gebaseerd is op scan chains en gebruikt wordt voor uitgebreide en snelle functionele testen van VLSI schakelingen. Het verwijderen of uitschakelen van de scan chain interface na de testen tijdens productie maakt dit nevenkanaal onbruikbaar voor aanvallers, maar verhindert ook het verder testen en updaten van de schakeling in een latere fase. In sommige toepassingen, zoals bijvoorbeeld set-top boxen, gebeurt het updaten van de firmware via de JTAG poort die intern is doorverbonden met de scan chains. In deze gevallen is het dus niet mogelijk de scan chains uit te schakelen en zal op een andere manier bescherming moeten gezocht worden tegen aanvallen. Daarenboven zal de kost in termen van oppervlakte en extra doorlooptijd van de testen minimaal moeten gehouden worden om deze beveiligingsmechanismen in te kunnen bouwen in een commercieel product. Dit werkstuk start vanuit een onderzoek naar de gevoeligheid zowel symmetrische-sleutel- als publieke-sleutel-implementaties in hardware t.o.v. scan-aanvallen. Daarna worden geschikte tegenmaatregelen voorgesteld om de bovenvermelde afweging tussen testbaarheid, beveiliging en kost aan te pakken.

Eerst wordt in deze thesis een beschrijving gegeven van scan-aanvallen op hardware-implementaties van het symmetrische-sleutel-algoritme AES en de publieke-sleutel-algoritmes RSA en ECC. Er wordt verondersteld dat er geavanceerde DfT structuren zoals test-compressie en X-handling in gebruik zijn. Daarnaast worden ook actuele tegenmaatregelen tegen nevenkanaalsaanvallen gebaseerd op vermogensanalyse en fout-aanvallen (fault attacks) geanalyseerd. Er wordt nagegaan of deze geschikt zijn om scan-aanvallen af te weren. Deze thesis evalueert ook de effectiviteit van verscheidene in de literatuur beschreven scan-aanval-tegenmaatregelen, zoals partiële scans en scan chain scrambling. Op het niveau van het algoritme, wordt aangetoond dat blinderen en schema’s gebaseerd op willekeur (“randomization schemes”) die bescherming bieden tegen differentiële vermogensanalyse ook veilig zijn tegen scan-aanvallen. Dit in tegenstelling tot tegenmaatregelen tegen eenvoudige vermogensanalyse en fout-aanvallen (fault attacks) waarvan aangetoond wordt dat ze geen bescherming bieden tegen scan-aanvallen. Op het RTL niveau wordt er aangetoond dat Multiple Input Signature Register (MISR) gebaseerde schema's inherent veilig zijn tegen scan-aanvallen, op voorwaarde dat enkel de laatste (en niet de tussentijdse) MISR handtekening observeerbaar is door de aanvaller. Ook worden er twee nieuwe tegenmaatregelen voorgesteld. Enerzijds op het systeemniveau: een veilige JTAG architectuur gebaseerd op het Schnorr protocol (gebruikmakend van ECC). Anderzijds op het poort-niveau ("gate level"): een ruis-injector die geïntegreerd werd met de test-compressie schema's.

Een volgende grote bijdrage van deze thesis zijn veilige cryptografische systeem-op-chip (SoC) testen. Als onderdeel van ons onderzoek werden scan-aanval-resistente Veilige Test Wrappers (VTW) ontworpen. Deze combineren een veilig authentiseringsmechanisme met de IEEE 1500 standaard Test Wrapper. Er worden twee varianten van de VTW voorgesteld: een eerste gebaseerd op het lichtgewicht blokcijfer KATAN en een tweede gebruikmakend van fysisch onkloonbare functies of PUF's (Physically Unclonable Functions). Hieraan gerelateerd werd ook gewerkt aan het integreren van efficiënte vermenigvuldiger-gebaseerde pseudo-willekeurige Logic Built-In Self-Test (LBIST) oplossingen met VTWs. Hiermee worden flexibele zelf-testen mogelijk voor SoCs met gewaarborgde beveiliging en beperkte kost op het vlak van tijd en oppervlakte.

Contents

3 Scan Attack Countermeasures 63

Part I: Countermeasures from other side-channel attacks

Part II: Countermeasures Specific for Scan Attacks

List of Figures

List of Tables

List of Abbreviations

AES	Advanced Encryption Standard
APUF	Arbiter PUF
ASIC	Application Specific Integrated Circuit
ATPG	Automatic Test Pattern Generation
BCU	BIST Control Unit
BIST	Built-In Self-Test
CA	Certification Authority
CRP	Challenge-Response Pair
CUT	Circuit Under Test
DES	Data Encryption Standard
DFT	Design For Testability
DPA	Differential Power Analaysis
DRM	Digital Rights Management
DSA	Differential Scan Attack
DUT	Device Under Test
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EDA	Electronic Design Automation
EDT	Embedded Deterministic Test
FA	Fault Attack
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
IP	Intellectual Property
HDL	Hardware Description Language

IWBR	Input Wrapper Boundary Register
JTAG	Joint Test Access Group
KFF	Key-dependent Flip-Flop
LFSR	Linear Feedback Shift Register
LUT	Look-Up Table
MISR	Multiple Input Signature Register
NIST	National Institute of Standards and Technology
OBU	On-Board Unit
OCSP	Online Certificate Status Protocol
OWBR	Output Wrapper Boundary Register
PKC	Public-Key Cryptography
PRPG	Pseudo-Random Number Generator
PUF	Physically Unclonable Function
RFID	Radio Frequency IDentification
RN	Random Number
RTL	Register Transfer Level
SA	Signature Analyzer
SCA	Side-Channel Attack
SHA	Secure Hash Algorithm
SKC	Secure JTAG Controller
SoC	System-on-Chip
SPA	Simple Power Analysis
STW	Secure Test Wrapper
SWIG	Simplified Wrapper and Interface Generator
TAP	Test Access Port
TAM	Test Access Mechanism
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select
TPG	Test Pattern Generator
TRNG	True Random Number Generator
TSC	Test Security Controller

Chapter 1

Introduction and Background

Cryptography is the science of secure communication and information protection from unauthorized access. Cryptography enables communicating parties to exchange information securely over an insecure channel. Modern cryptography not only applies to secure communication, but also has applications in software security, security of electronic devices (smart cards, RFIDs, memories, etc.), data protection (disk encryption), copyright protection (Digital Rights Management (DRM)) and more. Examples include RFID based access control systems, authenticating users for bank transactions using smart cards supporting cryptographic protocols, and full hard disk encryption employing symmetric-key cryptography. Cryptography as a technology can be used to provide the following security properties: confidentiality of data, data integrity, entity authentication of the sender and the receiver, and non-repudiation of sender and receiver, among others [89]. Various cryptographic primitives and protocols can help in attaining these objectives. Though the mathematical or theoretical strength of these primitives can be quite high, their implementations in hardware or software are prone to information leakages.

Cryptographic implementations in hardware and software need to be protected against attacks aimed at revealing the secret information stored within them. Hardware attacks can be characterized as follows:

- Active or passive attacks: active attacks require the attacker to tamper or perturbate the device internals (by probing, laser impingement, etc.) and derive the secret data from the observed response. Passive attacks require the attacker to only observe passively and infer the secret from the observed behavior by exploiting one or more physical characteristics of the device when it is in operation. Some of these characteristics include power consumption, electromagnetic radiation, execution timing, or the data coming in or out of the external interface.

- Invasive, semi-invasive or non-invasive attacks: invasive attacks require opening the device package and contact the electronic circuits inside; semi-invasive attacks also require opening the device package but no contact to the internal circuits is needed, while non-invasive attacks do not require modification of the device package.

The non-invasive and passive attacks are also termed as side-channel attacks where the physical characteristic opens a side-channel or backdoor through which secret information inside the cryptographic chips leaks. Some of these side-channels include electromagnetic radiation or power consumption of the circuit in operation.

In this thesis, we focus on one such side-channel which is the Design-for-Test (DfT) infrastructure incorporated in a circuit for thorough and rapid manufacturing test of the device. Scan chains are widely deployed in the semiconductor industry for structural testing after manufacturing or fabrication. These scan chains are internally connected to the IEEE Std 1149.1 JTAG (Joint Test Access Group) [6] Test Access Port (TAP) which provides the external interface to the tester/user. JTAG can be used in debuggers or for programming or updating devices, such as in set-top boxes for pay-TV subscriptions [5]. JTAG can can also be used for bitstream programming in FPGAs or for operating system upgrades in some smartphones.

Although scan chain DfT provides the highest testability, it can be used by an attacker to read chip-internal data, read stored secret information and determine the positions of all the secret elements in a chain. Scan chains may be permanently disabled after testing of the chip (by blowing some fuses, for instance) before being used in a product, but then the in-field testability of the chip is lost. Probing attacks can still be performed on parts of these broken scan chain elements and even the blown fuses can be carefully connected back. Moreover, the area cost to incorporate these fuses can be quite high. Hence, in this thesis, scan chains have been left intact for thorough testing of complex circuits, but at the same time protected against attacks exploiting the test infrastructure. The mechanism to achieve this dual paradigm of test and security should not add high area overhead or increase the test time to an unacceptable level. Hence, the purpose of this thesis is to design secure DfT structures which address the trade-off between security, test quality and test cost.

A basic introduction to testing and specifically structural testing is included here to initiate the reader to better appreciate the security challenges of the testing schemes. The two main DfT methods of using scan chains and Built-In Self Test (BIST) are briefly explained. This is followed by a discussion on modern DfT schemes including test compression logic and X-state handling techniques prevalent in state-of-the-art VLSI circuits.

After presenting the test schemes, scan-based side-channel attacks are introduced. The scan attack principle along with its assumptions and its two commonly occurring types – simple and differential scan attacks – are presented briefly. State-of-the-art differential scan attacks on symmetric and public-key cryptographic hardware implementations are introduced. Finally, a short discussion on test compression and its effect on scan attacks is given.

State-of-the-art scan attack countermeasures present in the literature are briefly introduced. This is followed by a short discussion on the countermeasures proposed in this thesis.

1.1 Introduction to Testing and Structural Testing

Testing of Integrated Circuits (ICs) is of crucial importance for the semiconductor industry. It ensures the quality of the product by determining if a circuit is working as intended. Due to the reduction of feature sizes in the VLSI era owing to Moore's law, the probability of occurrence of manufacturing defects resulting in a faulty circuit grows. Hence, efficient tests need to be designed to detect these faults, and also diagnose the cause of these silicon failures to increase yield and reduce costs. Testing digital ICs can be divided into two main categories: functional testing and structural testing.

1.1.1 Structural vs. Functional Testing

Functional testing consists of applying stimuli to the circuit interface in order to verify if the circuit behaves properly. However, it is not feasible to apply functional testing to large modern-day circuits, since testing all the possible stimuli and checking the corresponding responses may take exponential time. To further illustrate this, let us consider an example of testing a 10-input AND gate. In order to guarantee that the given circuit functions correctly as an AND gate, and not as any other gate (such as an OR, NOR or NAND gate), we need to check that the desired output is obtained for all possible 2^{10} (= 1024) input patterns. A complete functional test will check each entry of the truth table for these given inputs. Though possible, such a test will take too much time and would be impossible to use with a real circuit with several hundred input lines [23].

Therefore, the standard methodology for digital testing is the structural test. It is based on applying vectors that test all faults in the circuit, with gate-level precision. These vectors are generated based on the netlist simulation and using Automatic Test Pattern Generators (ATPG), like Synopsys' TetraMAX [11] or Cadence' Encounter True-Time ATPG [13]. Satisfying a structural test means that the circuit gates are working properly for the assumed fault models. If the circuit is properly designed, it also means that its functional behavior is as expected. In order to achieve high fault coverage during structural testing, the usual practice is to insert scan chains in the design. This is automatically done by transforming the flip-flops into scan flip-flops and connecting them in large shift registers. Besides the circuit input/output pins, the scan chain provides the tester an additional path to load input patterns (by shifting in vectors in the scan chain) and to unload response vectors (by shifting out). It is also possible to structurally test a circuit without employing full scan, however in some cases the fault coverage may be reduced. Structural test is usually performed at the fab after chip manufacture, using costly testers (Automatic Test Equipments (ATEs)).

1.1.2 Fault Models

The main advantage of structural testing is that it allows for the development of algorithms based on fault models. Most test generation and test evaluation (or fault simulation) algorithms are based on selected fault models [23]. The primary fault models are:

- Stuck at Faults: this fault is modeled by assigning a fixed (0 or 1) value to a signal line in the circuit. A signal line is an input or an output of a logic gate or a flip-flop. The most popular forms are the single stuck-at faults: stuck-at-1 (s-a-1 or sal) and stuck-at-0 (s-a-0 or sa0) [23]. This is the most popular fault model and many types of structural faults can be made equivalent to this model.
- Bridging Faults: a bridging fault is a short circuit between two or more cells or lines. It is a bidirectional fault, so either cell/line can affect the other cell/line. A 0 or 1 state of the coupling cell causes the fault, rather than a coupling cell transition [23].
- Delay Faults: these faults cause the combinational delay of a circuit to exceed the clock period. Specific delay faults are transition faults, gate-delay faults, line-delay faults, segment-delay faults, and path-delay faults [23].

1.1.3 Scan-Based Structural Design-for-Test (DfT)

Scan has been generally accepted as the standard method of testing chips due to high fault coverage and relatively lower area overhead. Inserting scan-chains while designing the chip requires a few additional/multiplexed pins to the primary inputs/outputs to serve as the scan-enable, scan-inputs and scan-outputs. Internally, there is little impact on the design since the standard flip-flops (FFs) are replaced by scan flip-flops (SFFs) (i.e., flip-flops with an input multiplexer) which are then linked to one another creating a shift register (scan chain). An example of a scan chain is shown in the Figure 1.1. Scan-enable selects between functional and test mode operations. It controls each multiplexer, choosing between the normal mode input of the FF or the output of the previous SFF in the chain.

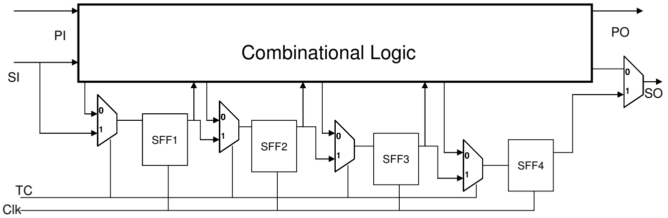


Figure 1.1: Scan chain DfT structure

Scan chains allow the tester to control and observe internal states of the circuit by loading/unloading input patterns/test responses. In order to load test patterns, the scan-enable signal is activated and each bit of the pattern is shifted in at each clock. When the entire input pattern is serially loaded, the scan-enable signal is deactivated for one or more cycles. During these cycles (capture mode), the input test patterns are applied to the combinatorial logic and the response is stored in the sequential elements. Scan-enable is activated again (shift-mode) and the internal state can be scanned out and be analyzed by the tester. At the same time the next input pattern is loaded. In other words, using scan chains essentially transforms the circuit into pure combinatorial logic, which is much easier to test than sequential logic.

Scan testing achieves the dual objective of testability – controllability and observability. Controllability is the ability to change the state of internal nodes using only the primary inputs, while observability means the ability to observe the state of internal nodes using only the primary outputs. Inserting scan-based DfT structures in a design allows the designer to achieve very high fault coverage using Automatic Test Pattern Generation (ATPG) tools during manufacturing tests.

1.1.4 Logic Built-In Self-Test (LBIST)

In Logic Built-In-Self-Test (LBIST), test patterns are generated internally by a Pseudo-Random Pattern Generator (PRPG) and the test responses are compacted in a Multiple Input Signature Register (MISR) to derive a signature of the circuit-under-test (CUT). This CUT signature is then compared with a stored golden signature of a fault-free circuit to take a pass/fail decision on the CUT. LBIST is normally used in conjunction with scan chains to achieve high testability. In scan-based LBIST, the PRPG is connected to the inputs of the scan chains and the MISR to the scan chain outputs. After the test stimuli have propagated through the logic, responses are captured into the registers and sent to the MISR. The BIST controller ensures that these steps are repeated until the test is finished [128]. A general block diagram of scan-based LBIST with its associated components is shown in Figure 1.2.

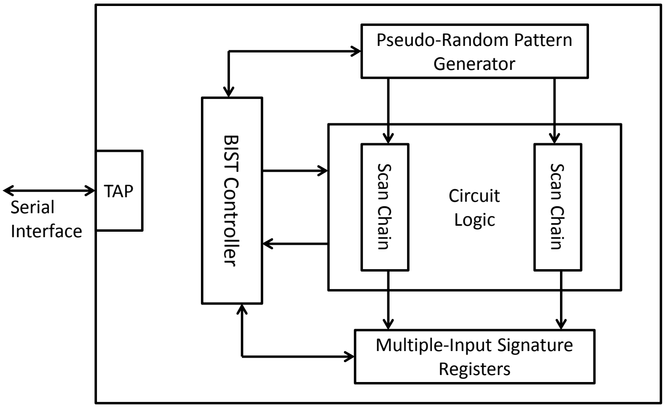


Figure 1.2: Scan-based Logic Built-In-Self-Test (LBIST) structure

LBIST provides inherent security, as the test response patterns are not available externally. However, it provides only a pass/fail decision which is not useful for diagnosis. It has relatively high area requirements when used in smaller ICs, since it needs the implementation of the PRPG, the MISR and the BIST controller, which are fixed-size components. In LBIST, the generated internal test patterns are pseudo-random and cannot be fully controlled.

1.1.5 Time Compaction

Time compaction uses sequential logic to compact test responses. Here, multiple input signature registers (MISRs) are employed to reduce test time. In order to minimize the need to shift out all test responses, the scan chain outputs are compressed into a signature with a MISR [128] (Figure 1.3). Similar to Built-in Self-Test(BIST), this compressed signature is compared with a golden signature to take a pass/fail decision on the device under test. Time compaction schemes also require special test structures to deal with unknown states (X-states) which can corrupt the BIST signature. These structures are elaborated in Section 1.2.2.

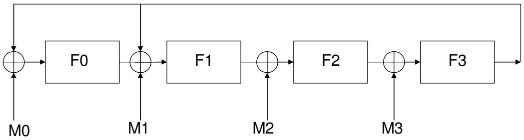
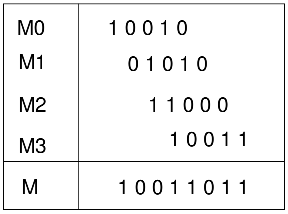


Figure 1.3: A four stage MISR



M0	1 0 0 1 0
M1	0 1 0 1 0
M2	1 1 0 0 0
M3	1 0 0 1 1
M	1 0 0 1 1 0 1 1

Figure 1.4: MISR Signature Calculation

1.1.6 Multiple Input Signature Register (MISR)

As mentioned in Section 1.1.4, MISRs are used in Logic BIST and time compaction schemes to obtain a compressed signature from the output responses of a CUT. MISRs consist of a series of flip-flops connected as a shift register with XOR gates in the forward or feedback paths. A four-stage MISR with feedback polynomial $f(x) = 1 + x + x^4$ is shown in Figure 1.3. Let the inputs on five consecutive clock cycles be M0=10010, M1=01010, M2=11000, M3=10011. Representing the inputs in polynomial form, $M0(x) = x + x^4$, $M1(x) = x + x^3$, $M2(x) = x^3 + x^4$, and $M3(x) = 1 + x + x^4$. As an example, let us take an effective input sequence polynomial as $M(x) = M0(x) + xM1(x) + x^2M2(x) + x^3M3(x)$. Thus, $M(x) = 1 + x^3 + x^4 + x^6 + x^7$, or 10011011. Hence, the signature R can be computed as the rightmost four bits of the $M(x)$. Therefore, R = 1011 [128]. This is represented in Figure 1.4. MISRs are similar to Linear Feedback Shift Registers (LFSRs) used in symmetric-key cryptography.

1.2 Test Compression

With a scan approach, the test time is proportional to the number of patterns and to the length of the scan chain. Thus, for large circuits with thousands of flip-flops, scanning in patterns and scanning out the responses may take too much tester time. This reflects on the final circuit cost. Therefore, it is a common practice to divide the flip-flops into multiple shorter scan chains, reducing the load/unload time. Additionally, to meet the constrained number of circuit pins, compaction structures are implemented.

Moreover, when test vectors are generated for a circuit by an automatic test pattern generator (ATPG), most of the test vector bits are unspecified, or don't care states, which are randomly filled with 0s or 1s, to enable their use on an ATE. These states can be removed from the test vectors in an efficient manner using test compression, allowing for a substantial reduction in test time and cost. The three popular industrial test compression tools are Synopsys' DFTMAX employing Adaptive Scan, Cadence' Encounter Test using OPMISR, and Mentor Graphics' Tessent TestKompress using Embedded Deterministic Test (EDT).

1.2.1 Space Compaction

These test structures consist of two parts: the decompressor that receives a small number of input patterns through the test inputs and spreads them over many scan chains, and the response compactor that combines all the scan chain outputs into a reduced set unloaded through the test outputs. An illustration of a Space Compaction scheme is shown in Figure 1.5. The figure presents a compression ratio of four, where two test inputs are spread into eight scan chains using an input decompressor and then the scan chain outputs are compacted into two test outputs using an output response compactor. The input decompressors may consist of demultiplexers or a combination of Ring Generators (which are variants of Linear Feedback Shift Registers) and Phase Shifters (which are used to spread the ring generator outputs in such a way so as to remove their mutual linear dependencies). The output response compactors are usually based on XOR-gate based parity trees. Space compaction schemes are normally combined with X-state handling schemes to deal with unknown states which can corrupt the test signature.

Commercial DfT tools available from various Electronic Design Automation (EDA) vendors employ different variants of space and time compaction. DFTMAX tool from Synopsys employs X-tolerant XOR-tree based space compaction in its Adaptive Scan approach. Cadence Encounter Test combines space compaction using static X-Masking with time compaction using MISRs in its OPMISR scheme. Mentor Graphics provides only XOR-tree based space compaction with dynamic X-Masking in its Tessent TestKompress tool using Embedded Deterministic Test (EDT) scheme.

1.2.2 X-State Handling

Compaction structures help reduce test time and cost by reducing the requirement on the number of tester pins and test patterns with minimal impact (around 1% [111]) on fault coverage. However, some flip-flops in the scan chain may depend on unpredictable values (e.g. previous states, memories, unknown bus values). These values are referred as unknown states (X-states). Therefore, in addition to providing space compaction, industrial DfT tools also have provisions for dealing

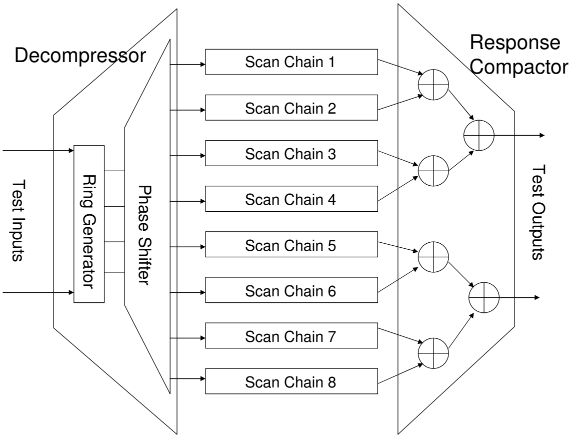


Figure 1.5: Generic Space Compaction

with these X-states. There are two popular methods used for achieving this: X-tolerance and X-masking.

1.2.2.1 X-Tolerant Schemes

X-tolerant methods are generally combined with logic BIST to prevent unknown values from corrupting the MISR signature. They allow test patterns to have any number of unknown values with no degradation in compression and application efficiency [131]. X-tolerant logic has the following generic structure. At the input side, there are decompressors to enable testing of multiple scan-chains using a reduced number of scan inputs. At the output side, there is an XOR network to connect multiple scan chains to fewer scan outputs in such a manner so as to counteract the effect of X-states. This way compression is achieved without compromising testability. Figure 1.6 shows the structure of a X-tolerant space compressor employed in Synopsys' Adaptive Scan.

Adaptive scan is the test compression architecture used in the Synopsys' DFTMAX test tool. At the input side, there are multiplexers (MUXes) to enable testing of multiple scan chains using a reduced number of scan inputs. At the output side, there is an XOR network to connect multiple scan chains to reduce the number of test outputs. The output side XOR compactor network is also known as the unload compressor: it provides an X-tolerant and low area overhead design. This helps in diagnosing high volume of scan pattern failures which can be observed on the tester [128].

The X-tolerant compressor [130] is based on an algorithm derived from Steiner

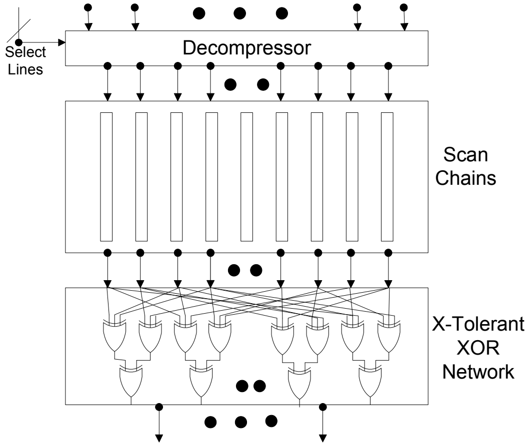


Figure 1.6: X-tolerant compressor logic structure

network trees [47]. Steiner trees calculate the shortest route for connecting all nodes in a network and are similar to minimum spanning trees. This special compressor structure provides a good balance between scan compression, X-tolerance and silicon area. To prevent aliasing, cancellation of simultaneous faults on two or more chains, a unique combination of sub-scan chains connects to each compactor output. This combination depends on the compaction structure employed. In some cases, the outputs are computed by XORing disjoint subsets of scan outputs.

1.2.2.2 X-masking Schemes

The presence of an X-state in one of the scan chains corrupts the test of the flip-flops in the other scan chains which are compacted by the test compression logic at the same clock edge. In order to avoid this situation, special structures (Masking logic in Figure 1.7) may be inserted to filter unknown values. Most techniques are based on having a mask input and a mask decoder that disables some chains in the presence of X-states, thereby masking their effects.

The mask can be static or dynamic. In static masking, a fixed mask value feeds the mask decoder, whereas in dynamic masking, part of the input vector is given to the decoder to generate a variable mask. A brief description of the commercial EDA test tools employing static and dynamic masking are given below:

- **Static X-masking** OPMISR (On-Product MISR) is one of the main features included in the Cadence Encounter Test toolkit [128]. The essential part of OPMISR is space compaction employing XOR trees. OPMISR has an optional feature of X-state handling using static X-masking. Time compaction

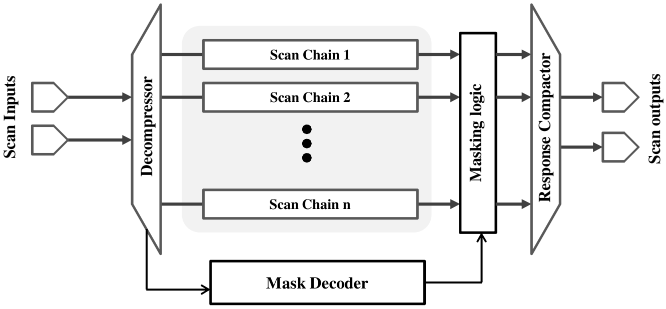


Figure 1.7: X-masking compressor logic structure

with Multiple Input Signature Registers (MISRs) is another optional feature provided by OPMISR. It is required for reduction of test time and thereby cost. Test outputs are generated faster depending on the desired compression ratio. Since the mask in static X-masking is fixed, the same scan chain(s) are masked for the entire test duration for a fixed input vector.

- **Dynamic X-masking** Mentor Graphics test compression tool, Tessent TestKompress employs Embedded Deterministic Test (EDT) [111, 110]. Similar to the test compression tools from Synopsys and Cadence, it uses XOR trees for space compaction. However, it deals with X-states in a different manner through the use of a dynamically changing mask, which provides more flexibility and is easy to apply, since knowledge of scan chains which have a higher probability of occurrence of X-states (as in static X-masking) is not required. X-masking is done with AND gates where the inputs are generated on-the-fly through a pattern mask decoder. The process of compaction begins by loading the pattern mask into a shift register that drives the selection logic. The mask is part of the input test patterns, and is loaded concurrently with the internal scan chains. The content of the pattern mask is compressed in the same manner as done for the test inputs [112]. The masking logic is dynamic and varies based on a special EDT clock and test inputs.

1.3 Introduction to Scan Attacks

Scan-chain DfT infrastructure is incorporated in a circuit for thorough structural testing. However, it can be targeted by attackers to extract secret information from security chips. The scan chains can act as a side-channel through which confidential information, such as secret keys stored inside the chip, can be recovered. As in other attacks exploiting side channels, knowledge of the underlying cryptographic implementation is required to derive useful information from the scan outputs.

1.3.1 Attack Principle

Though the scan-chain test approach provides the best controllability and observability to the test engineer, it can be used by an attacker to read chip internal data, to read stored secret information and to determine the position of all the scan elements in a chain. Only a few pins need to be externally controlled and monitored: the scan-in, the scan enable, and the scan output. By observing the scan chain output during a cryptographic process repeatedly, the secret information can be deduced through knowledge of the encryption algorithm. The ability of the circuit to switch between normal and test mode as well as the possibility to stop the execution of the cryptographic circuit at any time in test mode and scan out the responses is exploited in scan-based attacks.

Scan attacks can also be combined with other forms of side-channel attacks such as timing attacks or power analysis attacks to make them more powerful. For instance, the time instant at which the attacked circuit must be switched between functional and test modes may be inferred by observing the simple power analysis (SPA) traces of the circuit under operation. These traces can be used to determine the point at which an operation of interest completes execution (for example one square and multiply operation in the case of RSA or one point addition or doubling in the case of ECC).

1.3.2 Attack Target

The target of scan attacks is typically the register storing the computation results of intermediate operations. For instance, in case of AES, it is the round register, in the case of RSA, it is the intermediate register storing the result of each modular square and multiply operation, while in the case of ECC, it is the intermediate register storing the results of each point doubling or point addition operation. The flip-flops corresponding to the intermediate register are referred in this thesis as key-dependent flip-flops (KFFs). Scan attacks are possible even if the secret key register is not included in the scan chains.

1.3.3 Attacker Scenario and Assumptions

The following assumptions are made in the differential scan attack presented in this work.

- The scan enable pin can be controlled by the attacker.
- Details of the cryptographic algorithm are known to the attacker.

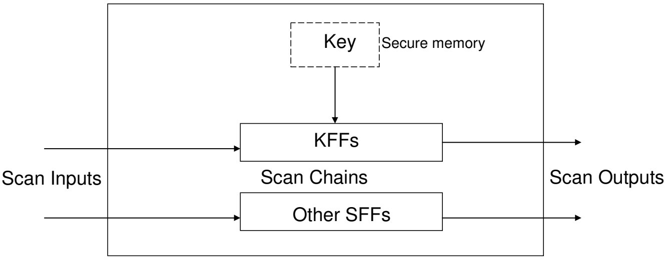


Figure 1.8: Simplistic Attacker Model

- The time required to execute the target operation is known to the attacker. In other words, the attacker can single-step the execution of the cryptographic algorithm.
- The test structure type (space compaction, time compaction, X-tolerance, X-masking) is known to the attacker. However, the attacker may not be aware of the exact details of the test scheme (such as the compression ratio or the number of scan chains in the DfT structure).

In addition to this, an implicit assumption in differential scan attacks is that all FFs of the design (except the KFFs) have the same value after one encryption step. This is an important requirement so that the differential process eliminates the effect of these key-independent FFs.

The basic attacker model employed in this work is represented in Figure 1.8. As mentioned in Section 1.3.2, KFFs represent the flip-flops of the intermediate register which are the target of scan attacks. The secret key is assumed to be securely stored in memory and not included on the scan chain. However, it is used to derive the contents of the KFFs. The other scan flip-flops (SFFs) of the design are assumed to contain values independent of the key. The attacker is assumed to have physical access to the circuit containing the scan-chain DfT, and hence can control and observe the scan chain contents through the external scan inputs and outputs. However, probing attacks on the chip internals after opening the device package is outside the scope of the attacks considered in this thesis.

1.3.4 Classical Differential Scan Attacks

The first scan attack proposed in the literature [132] was conceived to break a Data Encryption Standard (DES) implementation. Yang et al. describe a two phase procedure which consists in first finding the position of the intermediary registers on the scan chain, and then retrieving the DES first round key by applying only three chosen plaintexts.

In the first step, the scan-chain output is analyzed and the flip-flops corresponding to the round register are determined. This proceeds as follows: the chip is run in functional mode for one clock cycle. The result after the XOR of the key with the plaintext and the first round is stored in the round register. Then the chip is switched to scan mode and the contents of the bit stream are scanned out. This step is repeated for another plaintext input differing in one byte. The scanned out pattern is saved and the process is run for all 256 possible combinations of plaintexts.

The next step consists of finding the round key in a byte-by-byte manner. A chosen plaintext containing a particular byte is applied, and the corresponding word is observed on the round register. The corresponding ciphertext byte is determined and XORed with the plaintext byte to deduce one byte of the key. This process is repeated until all the bytes of the key are determined. This attack procedure is an implementation of the differential cryptanalysis principle as first introduced in [19].

1.3.5 Advanced Encryption Standard (AES)

AES is one of the widely used industrial standard block ciphers which encrypts blocks of 128-bit messages and supports variable key lengths: 128 bits, 192 bits, or 256 bits [39]. The AES round function consists of four operations: they are applied to the cipher state in the following order: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. The `SubBytes` operation is a non-linear transformation which operates on each byte of the state. `ShiftRows` rotates the bytes in each row of the state to introduce diffusion. The `MixColumns` is a linear operation that multiplies each column with a fixed polynomial over $GF(2^8)$ for achieving some further required diffusion. The `AddRoundKey` XORs the round key to the state. Further details regarding the specification of AES can be found in [39].

1.3.6 Modified Differential Scan Attacks

Modified Differential Scan Attacks (DSA) exploit the fact that two particular inputs to the round function of a block cipher can transform into output vectors with a unique Hamming distance after one round of encryption. For instance, if two plaintexts with an XOR difference of `0x01` in their least significant byte (LSB) are encrypted using only one round of AES, the Hamming distance between the output vectors after one round can only have a handful of values. The distribution of these values is given in Figure 1.9.

The `MixColumns` function of AES operates on columns of the state which can be defined as multiplying each column with a Maximum Distance Separable (MDS) matrix of branch number five. Therefore, any byte of the input will affect all

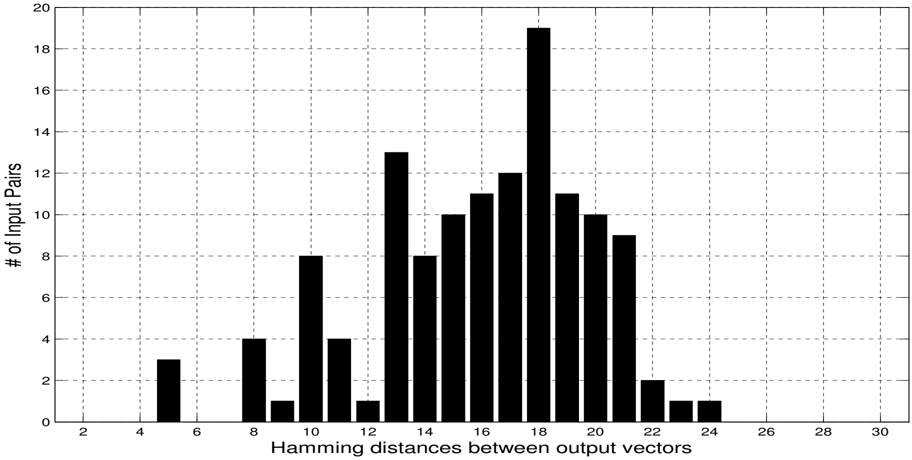


Figure 1.9: Distribution of Hamming distances, for `0x01` XOR difference on inputs [36]

four bytes of the output which forms the basis of differential scan-based attacks. The Branch Number of a linear transformation is a measure of its diffusion power. For instance, a non-zero byte difference in the first byte, as in Figure 1.10, will transform into a non-zero difference after `SubBytes` and will not be affected by the `ShiftRows` operation. However, the `MixColumns` operation will lead to four non-zero differences on that column. As the `AddRoundKey` operation only XORs the state with the round key generated by the key scheduling algorithm, it has no effect on difference propagation as long as the same key is used. Additionally, there is an initial key XOR step before the encryption starts, and this is the operation that we will target to recover the encryption key of AES.

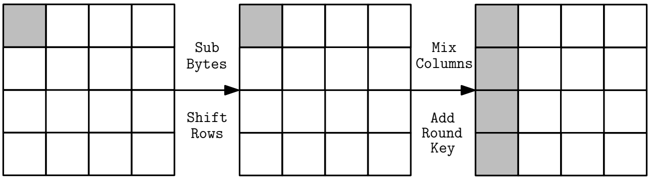


Figure 1.10: Difference propagation of a byte difference through an AES round

As pointed earlier, one byte difference in the plaintext will transform into a four byte difference due to the structure of the `MixColumns` operation. In fact, analysing the distribution of the Hamming distances for all 2^7 pairs generated with the byte difference `0x01` in their LSB, one can easily verify that there are four Hamming distance values (9, 12, 23 and 24) which can only be generated by a unique pair of inputs, as shown in Figure 1.9. Therefore, whenever such a Hamming distance is observed between the output vectors, one can XOR the corresponding plaintext byte with the pre-computed values to recover a byte of the encryption key.

As an example, let the attacker encrypt two plaintexts which have a difference of `0x01` in their LSB and observe a Hamming distance of 9 between the one round output vectors. Since the attacker can pre-compute the possible Hamming distances depending on all possible input pairs with `0x01` difference in their LSB, she finds out that the inputs to the S-box should either be `0xE2` or `0xE3`. Hence, the only thing that remains is to XOR these values to the corresponding byte of the plaintexts and obtain two possible keys, one of which is definitely the correct key byte. Proceeding in this way, the attacker can reduce the search space for the key from 2^{128} to 2^{16}, and eventually recover the 128-bit encryption key used in that AES implementation in negligible time. Here, the number of encryptions the attacker needs depends on the byte difference used for attacking the system. The more unique values in the Hamming distance distribution, the better the chances are for the attacker to be able to reduce the search space for that key byte.

Hence, differential scan attacks are a type of chosen plaintext attack. Similar to differential cryptanalysis, the statistical weaknesses in different ciphertext outputs are exploited when many pairs of plaintexts with the same input difference are applied.

The first scan attack on the Advanced Encryption Standard (AES) was proposed in [133]. It was based on the differential method, which analyzes the differences of scan contents instead of the direct value itself. By using this method, the preliminary step of identifying the position of the intermediary registers as in [132] is no longer required. Advances were also made on proving that public-key implementations are susceptible to scan attacks. RSA and Elliptic Curve Cryptography (ECC) keys are retrieved by methods described in [96, 94] respectively. Besides, some scan-attacks were also proposed for stream ciphers [82].

The simplicity of the attack enables it to be applicable to schemes with a simple compaction scheme such as an XOR tree compactor. However, this XOR operation can affect the observed Hamming distances depending on the distribution of KFFs over the scan chain structure. In [36, 35], Da Rolt et al. analyze the effect of an XOR compaction on the basic attack and show that the attack is still applicable to these systems.

Figure 1.11 shows a possible distribution of KFFs in the scan chains of a hardware design. As illustrated in the figure, a column of scan flip-flops containing one corresponding flip-flop for each scan-chain represents a slice. The flip-flops denoted by F_{ij} are ordinary scan flip-flops, whereas the flip-flops containing a key bit (KFF) are denoted by K_{ij}, where i stands for the scan-chain number and j indicates the position of the respective flip-flop in the scan-chain. Any slice containing one or more KFFs is called an active slice, while the others are called non-active slices.

For instance in the figure, the slice containing KFFs K_{42} and K_{82} represents an active slice. Similarly, a scan chain containing one or more KFFs is denoted as an active scan chain, while the other scan chains are called non-active scan chains.

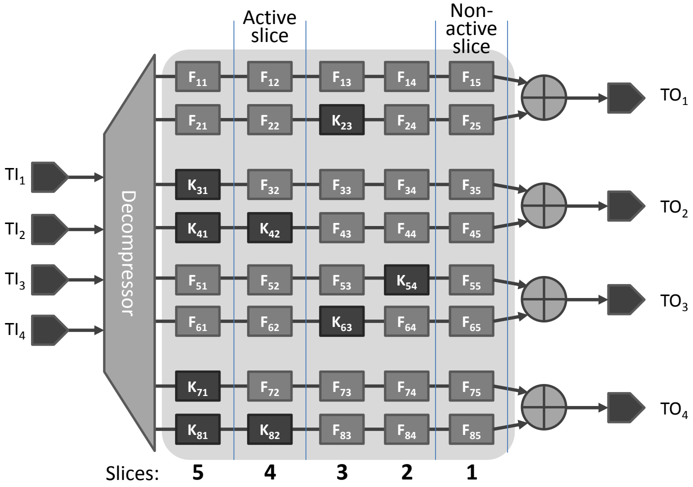


Figure 1.11: Slices and active slices

In this thesis, differential scan attacks on symmetric-key and public-key cryptographic hardware implementations are performed for various distributions of active scan chains and active slices along with different distributions of KFFs on the active scan chains and slices. The attacks are repeated for a large number of times to ensure repeatability of results. This gives a fair indication of the success probabilities of the attack when the KFFs are spread randomly over the scan structure, as is possible in a realistic industrial scenario. Various commercial test compression structures including XOR-tree based space compression, X-masking, X-tolerance, MISR based time compression, and modified scan structures claimed to act as countermeasures such as inverted scan chains, partial scan and scan chain scrambling schemes, are subjected to these scan attacks. Requirements on the number of input messages or points for a successful scan attack are also presented along with procedures to reduce the numbers. Detailed scan attack success results are presented in order to draw an informed conclusion to help the test engineer decide on a suitable secure DfT structure.

1.3.7 Recent Differential Scan Attack

In [37], a new method is proposed to perform DSA on AES circuits exploiting the linear structure of the `MixColumns` operation in AES. Here, the authors proposed to look for certain differences after the `SubBytes` operation rather than providing two plaintexts with a certain Hamming difference in between as with earlier DSA approaches. This attack is shown effective even when X-masking schemes, Partial Scan and MISR based time compaction are present.

This attack focuses on obtaining a specific difference at the S-box outputs of the first AES round after the non-linear substitution operation. Rather than encrypting two plaintexts with a certain byte difference in between, one can also generate plaintext pairs which give a certain difference after the `SubBytes` operation, for a given (or guessed) key. If a fixed difference can be achieved after the S-box, the linear structure of the `MixColumns` operation will always distribute the difference over the state in the same way. In other words, the XOR of the first round outputs corresponding to the given plaintext pair is always a fixed value if the key guess is correct. Therefore, as long as the same test parameters can be set, the XOR difference of the test outputs should be exactly the same, leading to much more powerful attacks than the earlier efforts.

Since all the other operations of the AES round as well as the test compression schemes are linear, it is possible to make an exhaustive search in offline tables constructed for all possible cases for a particular difference at the S-box outputs. This is in contrast to the previous attacks in [133] where the Hamming distance between pairs of ciphertexts is calculated and investigated only after one complete round of encryption, when pairs of plaintexts with one byte difference are input to the circuit.

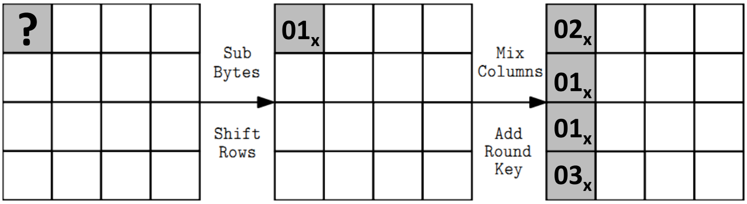


Figure 1.12: Recent differential scan attack representation

The attack in [37] consists of identifying a leaking bit followed by retrieving the secret key. The process for the retrieval of the secret key is represented graphically in Figure 1.12. It consists of the following steps: first, a key guess is made. Then a set of pairs which should give a one-bit difference after the `SubBytes` operation of AES is compiled. This gives some fixed values after the MixColumns operation due to the linearity property of the operation. This is followed by collecting the outputs. Finally, a check is made for a particular difference.

This attack can target even advanced DfT features such as X-masking. It is based on the reasoning that the mask decoder in the case of X-masking schemes can be controlled by setting the inputs in an appropriate way, as is generally the case with most commercial EDA test tools. Hence, the inputs can be changed continuously until one of the KFFs present on the scan chain is unmasked and appears at the scan outputs. Even if one key-dependent flip-flop (KFF) is unmasked or present on the scan chain, the attack can successfully recover the secret key.

1.3.8 Test Compression and security

Since in test compression schemes, the externally observable values are only the compressed stimuli and compacted responses, one would expect them to have some security properties as well. The theoretical security analysis of one of the popular test compression schemes from Mentor Graphics tool, Embedded Deterministic Test, was presented in [81], while the security claims of Tessent TestKompress are presented in a Mentor Graphics whitepaper [90]. Although no such security claims have been made to-date on the test compression tools from Cadence and Synopsys, they offer similar test response compaction structures. Moreover, in the X-state handling schemes of these industrial DfT tools, due to the combination of different scan chain outputs at the response compactor, there is further loss of observability of the internal states of the scan chains compared to a simple XOR tree compactor, which makes scan attacks somewhat more complicated.

1.4 Scan attack countermeasures

Various countermeasures have been proposed in the literature to counteract scan attacks. They can be classified into the following groups: structures that are offered by commercial DfT tools and may be considered as inherent countermeasures (such as test compression, X-tolerance, X-masking, MISRs); protocol countermeasures which change the test procedure in order to secure the test access (such as secure JTAG [104], or using secure mode for test access control); scan chain structure modification (such as unbounding, scrambling [63]), and countermeasures that resist against micro-probing of the scan signals (intrusion detection mechanisms by raising alarms in the case of unauthorized test access [62], or using spy flip-flops).

In this thesis, several existing countermeasures are evaluated and new countermeasures are proposed at various abstraction levels, from protocol level, algorithmic, system level, SoC-level to gate-level countermeasures. State-of-the-art side-channel countermeasures effective against power analysis and fault attacks are studied to investigate their possible reuse in protecting against scan attacks. Broad guidelines are also provided to help the test engineer design the cryptographic system in a manner so that it has inherently high resistance against scan attacks. The proposed countermeasures are designed to not affect the test quality or test time in an adverse manner and also without adding substantial area overhead of the security mechanism. Thus, they address the trade-off between testability, security and test cost to a large extent.

A cost-benefit analysis is also performed for the existing and proposed scan attack countermeasures in the thesis. Scan attack countermeasures always add some area and timing costs, but provide security to the test infrastructure of a circuit or system. Since the countermeasures are designed on various hardware platforms, it is not possible to compare them in absolute terms. However, area and timing costs for existing countermeasures are reported in this thesis wherever available. For the newly proposed countermeasures, detailed area results (for ASIC and FPGA platforms) and timing costs (in terms of number of clock cycles) along with the security claims are presented to help a secure systems designer choose the right countermeasure for an intended application.

1.5 Summary of Contributions

In this section, the structure of the thesis is discussed along with the personal contributions in the chapters.

- Chapter 1. This first chapter gives an introduction to testing, design-for-test, test compression and scan attacks. It provides the background to understand the rest of the thesis. Part of the material is derived from published and submitted papers indicated at the beginning of the thesis.
- Chapter 2. This chapter presents differential scan attacks on symmetric-key (AES) and public-key (RSA, ECC) cryptographic hardware implementations. Some modern DfT structures such as test compression, X-tolerance and X-masking are also investigated to study their security properties when applied to cryptographic circuits. The material on the scan attack on AES presented in this chapter is based on the DSD 2012 paper [49] and the accepted IEEE TCAD journal paper [41]. The material on the scan attack on public-key cryptographic hardware implementations is based on the published paper in COSADE 2012 [33] and in DFT 2012 [32].
- Chapter 3. In the first part of the third chapter, differential scan attacks on cryptographic hardware implementations in the presence of side-channel countermeasures are presented. Simple Power Analysis (SPA), Differential Power Analysis (DPA) and Fault Attack countermeasures are evaluated to determine their suitability to protect against scan-based side-channel attacks. This can help achieve reuse of existing side-channel countermeasures to protect against another side-channel attack, thus reducing overhead costs. This is based on the published JCEN 2012 journal paper [34]. In the second part, past work on scan attack countermeasures are described along with an analysis

of their effectiveness and cost. Scan attack on combined test compression and scan attack countermeasures is also presented which is derived partly from the TRUDEVICE 2013 workshop paper [48]. A gate level noise injector countermeasure augmenting test compression schemes is proposed. This is derived from the accepted IEEE TCAD journal paper [41].

- Chapter 4. This chapter presents contributions towards designing a secure test infrastructure by presenting countermeasures at the system and SoC levels. System level countermeasures in the form of a secure JTAG implementation are first presented. This is based on the published JETTA 2013 journal paper [40]. SoC-level countermeasures in the form of secure Test Wrappers are also proposed. These are derived from the ETS 2011 paper [42] and from the DATE 2012 paper [43]. Another SoC countermeasure employing a secure BIST solution is presented which is based on the technical report on secure mutual testing of cryptographic cores [71].
- Chapter 5. In the final chapter, the thesis is concluded with a discussion on future work. A summary of the results for differential scan attacks and countermeasures is also presented.

Chapter 2

Differential Scan Attacks on Advanced DfT Schemes

Publication Data

The material in this chapter is based on the following publications:

[A] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, “Security Analysis of Industrial Test Compression Schemes,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2013, vol. 33, no. 12, 2013.

[B] B. Ege, A. Das, S. Ghosh, and I. Verbauwhede, “Differential scan attack on AES with X-Tolerant and X-Masked Test Response Compactor,” Proceedings of the 15th IEEE Euromicro Conference on Digital System Design - DSD 2012, IEEE, pp. 545-552, 2012.

[C] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “A new scan-attack on RSA in presence of industrial countermeasures,” Proceedings of the 3rd International Workshop on Constructive Side-Channel Analysis and Design - COSADE 2012, Springer LNCS vol. 7275, pp. 89-104, 2012.

[D] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “A new scan attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures,” Proceedings of the IEEE International Symposium of Defect and Fault Tolerance in VLSI and Nanotechnology Systems - DFT 2012, IEEE, pp. 43-48, 2012.

Section 2.2 is primarily based on [A] with minor additions from [B]. Section 2.3.1 is based on [C], while Section 2.3.2 is based on [D].

Personal Contributions

- One of the principal authors.

2.1 Introduction

In the first part of this chapter, scan-based side-channel attacks are presented on symmetric-key and public-key cryptographic hardware implementations containing popular DfT structures generated by the major test tools of leading EDA vendors: Synopsys, Cadence, and Mentor Graphics. The test structures specific to these tools are incorporated on the cryptographic circuits using the DfT toolkits. Success rates of the attacks are reported for different DfT configurations. Though AES is taken as a case study, the scan attack principle outlined in this work is also applicable for other block ciphers which have similar diffusion properties.

In the latter part of the chapter, RSA and ECC public-key cryptographic hardware implementations containing test compression and other advanced DfT features are subjected to scan-based side-channel attacks and the experimental results are given. Requirement on the number of input messages/points for different configurations of key-dependent flip-flops and DfT structures are also presented.

2.2 Differential Scan Attacks on Symmetric-Key Implementations

2.2.1 Attack Strategy

The differential scan attack (DSA) outlined in this chapter is performed on software emulations of the DfT structures. The basic attack approach is presented in Figure 2.1. It is divided into two phases: analysis in hardware on the actual AES design and analysis in software comprising the attack. In the hardware analysis phase, the test structures of industrial DfT solutions are derived by inserting DfT functionality to the AES design using the DfT tools. This is followed by deriving possible inputs to the round function of AES for corresponding input differences in one byte. In the software analysis phase, the scan attack is performed by emulating the DfT structures in a C program, making use of the XOR differences and possible inputs derived in the online phase. An attack is deemed to be successful whenever

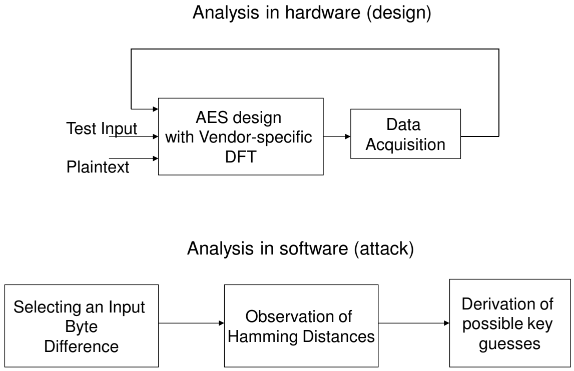


Figure 2.1: Online data acquisition from hardware and offline analysis in software

the correct key byte is suggested as the most likely key byte after the attack. Success rates are computed for 10000 random permutations of KFFs for each unique combination of active slices and active scan chains. The randomization is done in software using the C function rand(), with the seed value set to the current timestamp using the srand() function.

In this work, differential cryptanalysis principles are used for the scan attacks on cryptographic hardware implementations to deal with the obscurity introduced by advanced DfT structures. Due to the compaction of the scan outputs by the XOR compactor and further loss of observability introduced by X-handling schemes, linear cryptanalysis would have required much higher effort to achieve similar attack rates compared to differential cryptanalysis, and hence is not applied. Moreover, employing differential analysis helps remove the effect of other input dependent flip-flops in the scan chains.

In this work, we have used an open-source AES implementation from the Gezel hardware/software co-design website [3] as our target implementation. This design contains a table-lookup based S-box. The Gezel HDL code has been converted into VHDL using the `fdlvhd` converter tool and synthesized into a gate-level Verilog netlist using the Synopsys Design Compiler v2009.06 with a Faraday 130 nm library, on which the test compression structures are added.

Through software simulations, we made further observations on the effects of various Hamming distances between input plaintext pairs, other than the ones given in previous works [36, 35]. We used a C program to verify the attack given in [36], and observed that there are seven more one-byte differences (`0x89`, `0x92`, `0xE7`, `0x4A`, `0x69`, `0x6B` and `0xF5`) that result in four unique Hamming distance values after one round of encryption. We further observed that using difference `0xD1` results in five unique Hamming differences which can be used to infer a byte of the encryption key. The differences and the corresponding unique Hamming distance values used in this work are given in Table 2.1.

Table 2.1: XOR differences and corresponding unique Hamming distances after one round AES encryption

XOR difference	Unique Hamming distances
`0x01, 0x89, 0x92, 0xE7`	$\{9, 12, 23, 24\}$
`0xD1`	$\{5, 9, 12, 23, 24\}$
`0x4A, 0x69, 0x6B, 0xF5`	$\{8, 9, 12, 24\}$

All the attack success rate results presented in the following sections are obtained employing attack code written in C customized for the specific test compression structure or scan attack countermeasures. Simulations are performed on a 64-bit x86-64 Intel Core i7-2600 CPU running at 3.4 GHz having seven virtual processors and 8GB of RAM.

2.2.2 Distributions Considered

The AES round register, which is the target of scan attacks, can be spread in the scan structure in several ways unknown to the attacker. Our differential scan attack considers the following distributions of active slices and active scan chains to cover some typical scenarios.

- 32 active scan chains with active slices varying from 16 to 32.
- 32 active slices with active scan chains varying from 16 to 32.
- 24 active scan chains with 24 active slices.

After one round of AES, one key byte can affect 32 bits of the round register. Therefore, in the scan attack, if we consider one key byte at a time, then at most 32 key-dependent flip-flops would be in the scan structure at one time. This indicates that there will be at most 32 active scan chains when each scan chain contains one key-dependent flip-flop (KFF). Similarly, there will be at most 32 active slices when each slice contains one key-dependent flip-flop. These represent two extreme cases. To consider other practical scenarios, the number of active scan chains are varied from 32 to 16 keeping active slices fixed at 32, and vice-versa. An intermediate scenario of 24 active scan chains and 24 active slices is also considered to show the effect of different distributions of KFFs even when the number of active slices and active scan chains is fixed. Table 2.2 shows the distributions used in this work. Though statistically some distributions may occur more frequently than others, we did not take this into account.

Table 2.2: Distributions for 24 active slices / scan chains

#	Distribution
1	$\{2,2,2,2,2,2,2,2,1,\ldots,1\}$
2	$\{3,2,2,2,2,2,2,1,\ldots,1\}$
3	$\{3,3,2,2,2,2,1,\ldots,1\}$
4	$\{3,3,3,2,2,1,\ldots,1\}$
5	$\{3,3,3,3,1,\ldots,1\}$
6	$\{4,2,2,2,2,2,1,\ldots,1\}$
7	$\{4,3,2,2,2,1,\ldots,1\}$
8	$\{4,3,3,2,1,\ldots,1\}$
9	$\{4,4,2,2,1,\ldots,1\}$
10	$\{4,4,3,1,\ldots,1\}$
11	$\{5,2,2,2,2,1,\ldots,1\}$
12	$\{5,3,2,2,1,\ldots,1\}$
13	$\{5,3,3,1,\ldots,1\}$
14	$\{5,4,2,1,\ldots,1\}$
15	$\{5,5,1,\ldots,1\}$
16	$\{6,2,2,2,1,\ldots,1\}$
17	$\{6,3,2,1,\ldots,1\}$
18	$\{6,4,1,\ldots,1\}$
19	$\{7,2,2,1,\ldots,1\}$
20	$\{7,3,1,\ldots,1\}$
21	$\{8,2,1,\ldots,1\}$
22	$\{9,1,\ldots,1\}$

Even if AES forms a small part of an industrial SoC consisting of millions of gates, the distributions considered are still applicable. The reason behind this is that independent of the size of the design, the maximum number of active slices or active scan chains is always equal to 32 (as only 32 FFs would be key-dependent and the rest unrelated when subjected to differential scan attacks).

In Table 2.2, each digit under ‘Distribution’ represents the number of KFFs in each active slice/scan chain. There are 24 KFFs in total in each distribution. For instance, the first distribution has two KFFs on 8 slices/scan chains, and one KFF on another 8 slices/scan chains. Similarly, the second distribution has 3 KFFs on one slice/scan chain, two KFFs on 6 slices/scan chains, and the remaining 9 KFFs one each on 9 slices/scan chains. For the 22nd distribution, there are 9 KFFs on one slice/scan chain and one KFF each on 15 slices/scan chains. Since there are 22 possible choices for distributing active slices/scan chains, there are a total of $22^2 = 484$ distributions with 24 active slices and 24 active scan chains. These distributions are in fact all possible choices to distribute 32 KFFs over 24 active slices/scan chains. Since in these distributions, there are multiple FFs on each active slice, we consider cases where one KFF would mask another on the same slice, as is possible in a realistic scenario.

2.2.3 Scan Attack on AES Hardware in the Presence of XOR compaction with X-Tolerance

As explained in Section 1.2.2.1, the X-Tolerant XOR compressor used in Synopsys Adaptive Scan has a different structure than an XOR-tree. In an XOR-tree, each input contributes only once to the compactor outputs, whereas in the Adaptive scan structure, an input can be XORed more than once to derive the compactor outputs to prevent X-state propagation. Hence, DSA success rates are expected to be different as there is further loss of observability when DSA is concerned. Observed Hamming distances (HD_{obs}) vary from the actual Hamming distances (HD_{Real}) depending on the structure of the XOR network, therefore providing some security through obscurity as long as the structure of the XOR connections is not known. However, this compressor also leaks some information on the Hamming distance between outputs as it consists of linear operations.

Description of the Attack

To simulate a realistic scenario where the structure of the X-Tolerant logic is unknown to the attacker, we apply a generic attack independent of the compactor structure. Five XOR differences are used to amplify the effect of correct key guesses, and results are presented in Table 2.3.

Similar to the initial attack in [134], the attack consists of two main steps. First, all 256 possible values are given to the first byte of the plaintext and corresponding one round outputs are collected. In the second step, these outputs are paired depending on the selected input difference and the Hamming distance between these two selected outputs are computed. Unlike previous works, we use five different XOR differences (namely `0xD1,0x01,0x89,0x4A,0x69`) to amplify the visibility of the correct key among other key guesses. Note that all test outputs are XORed together to obtain a single value for the Hamming distance. This is important since the structure of the X-Tolerant logic used can vary depending on the number of scan chains and the number of test outputs in the design. The X-Tolerant logic used in this work, which has been derived from an actual test compression DfT insertion with a 32:8 compressor on the AES design by Synopsys' DfT Compiler, combines the scan chain outputs in the following way:

$$
\begin{aligned}
out_0 &= s_2 \oplus s_5 \oplus s_{24} \oplus s_{26} \oplus s_{19} \oplus s_{13} \oplus s_8 \oplus s_{29} \oplus s_{16} \odot s_{22} \oplus s_0 \oplus s_{11}\\
out_1 &= s_3 \oplus s_8 \oplus s_5 \oplus s_{27} \oplus s_{16} \oplus s_{18} \oplus s_{13} \oplus s_{21} \oplus s_{23} \oplus s_{29} \oplus s_0 \oplus s_{10}\\
out_2 &= s_{28} \oplus s_{25} \oplus s_{17} \oplus s_{20} \oplus s_{22} \oplus s_{14} \oplus s_9 \oplus s_3 \oplus s_{31} \odot s_6 \oplus s_0 \odot s_{11}\\
out_3 &= \neg s_{30} \oplus s_{27} \oplus s_{14} \oplus s_{19} \oplus s_{21} \oplus s_1 \oplus s_6 \oplus s_{22} \oplus s_3 \oplus s_8 \oplus s_{11} \oplus s_{17}\\
out_4 &= s_1 \odot s_9 \oplus s_{12} \oplus s_{15} \oplus s_{18} \oplus s_4 \oplus s_{26} \oplus s_{20} \oplus s_{31} \odot s_6 \oplus s_{23} \oplus s_{29}\\
out_5 &= \neg s_{30} \oplus s_{27} \oplus s_{14} \oplus s_{19} \oplus s_{25} \oplus s_{12} \oplus s_7 \oplus s_4 \oplus s_1 \oplus s_9 \oplus s_{16} \odot s_{22}\\
out_6 &= s_{10} \oplus s_7 \oplus s_{13} \oplus s_{21} \oplus s_{28} \oplus s_{15} \oplus s_{31} \oplus s_2 \oplus s_{24} \oplus s_{26} \oplus s_{18} \oplus s_4
\end{aligned}
$$

$out_7 = s_{28} \oplus s_{25} \oplus s_{17} \oplus s_{20} \oplus \neg s_{30} \odot s_{23} \odot s_{12} \oplus s_{15} \oplus s_2 \oplus s_5 \oplus s_{10} \oplus s_7$
where $s_i,\ i \in \{0, \ldots, 31\}$ are the scan chain outputs, while $\oplus$ and $\odot$ indicate XOR and XNOR operations respectively.

In the compactor structure above, each scan chain occurs at least twice at the compactor outputs to satisfy the X-Tolerant requirement of canceling X-states occurring on the scan chains. Since all the scan chains are included more than once, evaluating compactor outputs separately can result in an incorrect estimation of the actual Hamming distance between the corresponding scan designs. Therefore, the test outputs are XORed to make sure that the observed Hamming distance is smaller than or equal to the actual Hamming distance, and in fact never a larger value. When this approach is not followed, it becomes more difficult to correctly estimate the actual Hamming distance. This is because without the knowledge of the exact structure of the output compactor, it would not be possible to tell which scan chains contribute to the differences observed in particular compactor outputs. In some cases that we observed for different X-Tolerant logic designs generated by Synopsys, a scan chain is included in an even number of compactor outputs. In this case, XORing the compactor outputs will cancel out these scan chain outputs, therefore degrading the observability of the actual Hamming distance between two scan designs. When the case in the example above is considered, one can easily see that s_{22} and s_{24} are the only ones that are included in an even number of times in the compactor output. Therefore, when performing the analysis, they will be canceled out and therefore information about two particular scan chains s_{22} and s_{24} will be lost. Hence, a decrease in the success rate for a random distribution of KFFs is to be expected when compared to a simple XOR-tree compactor structure.

The attack methodology is slightly different from the conventional scan attacks. A key guess is performed if the observed Hamming distance is one of the extreme cases. For example for the XOR difference `0xD1`, we make a key guess if the observed Hamming distance is less than 5 or 9, and if it is exactly equal to 23 or 24. Our experiments show that this approach improves the overall success rate of the attack.

Attack Results

Table 2.3 summarizes the attack success for different number of active slices and active scan chains. The results suggest that the attack success decreases with decrease in the number of active slices. However, it seems that the success rate of the attack is affected to a lesser extent by variations in the number of active scan chains.

Figure 2.2 tells another interesting story. Although the number of active slices affects the success rate quite significantly (see Table 2.3), it seems the effect of the distribution over 24 active slices is minimal. When the number of active slices and

Table 2.3: DSA Success rates for X-Tolerant Logic for different distributions

#Active Scan Chains = 32		**#Active Slices = 32**	
#Active Slices	**Success Rate**	**#Active Scan Chains**	**Success Rate**
32	74.94%	32	74.94%
31	71.22%	31	74.83%
30	70.24%	30	74.24%
29	66.35%	29	74.11%
28	62.65%	28	74.28%
27	60.60%	27	74.45%
26	59.16%	26	74.22%
25	57.93%	25	74.13%
24	57.26%	24	74.06%
23	55.90%	23	74.19%
22	54.38%	22	74.56%
21	49.65%	21	73.58%
20	50.66%	20	61.98%
19	49.79%	19	56.65%
18	44.62%	18	56.78%
17	37.59%	17	57.34%
16	30.26%	16	56.09%

the number of active scan chains are fixed at both 24, the average success rate of the attack is around 76.01% for a random permutation of KFFs (i.e. at each of the 10000 runs, a random collection of KFFs are masked out because of the attack approach). Note that with the X-Tolerant logic structure used in this work (32 to 8 output compaction), the information on only two scan chains (namely s_{22} and s_{24}) are lost after XORing the two test outputs. Although the loss of information is minimal, some distributions of scan chains lead to further reductions in the observed Hamming distance value, thereby, degrading the attack success rate. Experiments are repeated for 10000 random distributions of KFFs over the whole design to have reliable statistics on the attack success.

It takes an average of 1.28 milliseconds to perform the attack once in software, using the configuration mentioned in Section 2.2.1 on a design with a random distribution of KFFs.

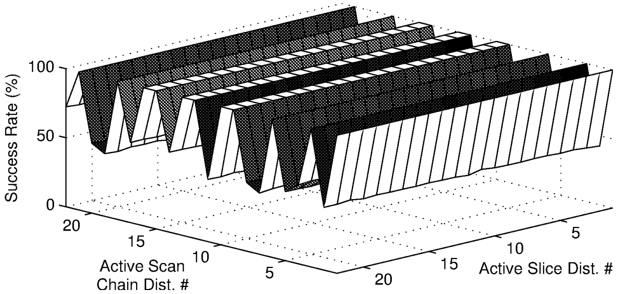


Figure 2.2: Success rate of the attack on Adaptive Scan for 24 active scan chains and 24 active slices (Distributions from Table 2.2)

2.2.4 Scan Attack on AES Hardware in the Presence of XOR Compaction with Static X-Masking

The space compactor of Cadence OPMISR is an XOR-tree against which differential scan attack in [35] is effective. As mentioned in Chapter 1, some other features of Cadence OPMISR are X-Masking provided for preventing X-states from corrupting the compacted test outputs, and MISRs for time compaction. However, they can also enhance security by reducing the observability of the internal registers in a design.

In this thesis, results of scan attack for a wide variation of active slices and active scan chains are presented for two cases: XOR-tree with static X-Masking, and for combined XOR-tree, static X-Masking and MISR structures.

2.2.4.1 Description of the Attack

Unlike in Section 2.2.3, the attack on static X-Masking is mounted based on the assumption that there are different randomly generated masks for different test inputs. Therefore, by repeating the attack for different test inputs, one can get a better idea of the actual Hamming distance between the first round outputs. In Figure 2.3, the trend in success rate in comparison to the number of test inputs used are presented. It can be observed from the figure that the attacks on masking schemes are more successful if a larger collection of random test inputs are used to mount an attack.

As in Section 2.2.3, the attack consists of two stages. First, all 256 possible values are given to the first byte of the plaintext and the corresponding test outputs are collected. In the second stage, the test outputs are paired depending on the chosen XOR difference and a key guess is made depending on the Hamming distance between output pairs. This two-stage attack is repeated multiple times for different mask values. This process continues till the hit counts for the correct key guess becomes greater than the incorrect key guesses, and becomes the top key candidate among the ordering of possible key bytes. Therefore, an attack is deemed successful only if the top key candidate is the correct one. In case this is not the case, the attack is not successful for the particular combination of input and mask values.

2.2.4.2 DSA on XOR Compaction with Static X-Masking

Static masking can have a significant impact on the differential scan attack especially for big designs, where there are more scan chains than the number of KFFs in the design. For AES, the attack complexity is upper bounded by 2^{32} as there can be at most 32 KFFs for a one-byte input difference. Therefore, recovering the encryption key of any AES design with more than 32 scan chains requires the same amount of effort as with 32 scan chains. The main reason behind this is that, with static masking, an entire scan chain is excluded for the whole testing process. This leads to having a particular scan chain contributing to the test output with probability $\frac{1}{2}$ (i.e., either the scan chain is included or not). Hence, assuming the worst case scenario in which the KFFs are being spread over 32 scan chains, the probability of all these scan chains contributing to the test output is $\frac{1}{2^{32}}$.

The attack method that we used is composed of two main parts: data acquisition and analysis. Data acquisition phase of the attack can be summarized as follows:

- Pick a test input at random
- Switch to test mode and fill the scan chain structure with the test data generated by the test input.
- Switch to functional mode and input a random plaintext to AES.
- Let the AES encryption run for only one clock cycle.
- Switch to test mode again, and obtain the test output for that particular input.
- Repeat the above steps for the same test input and with plaintexts which only differ in one byte (note that there are 256 such plaintexts)

The analysis phase of the attack can be briefly summarized as follows:

- Pair the test outputs so that their corresponding plaintext inputs have `0xD1` difference on their LSB, and `0x00` difference on all other bytes.
- XOR each pair together to get the observed Hamming distance HD_{obs}.

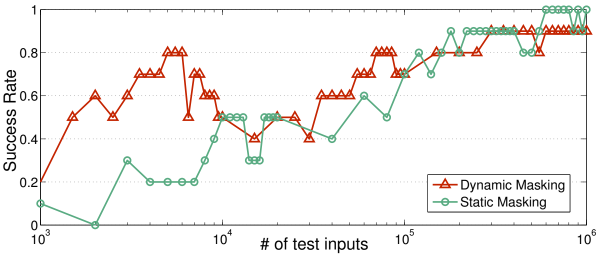


Figure 2.3: Change in success rate with respect to # of test inputs (or masks) used for the attack

- If HD_{obs} is odd and greater than 5, then assume HD_{Real} is either 9 or 23 and XOR the corresponding precomputed input values with the plaintexts to get possible keys. Similarly, if HD_{obs} is odd and greater than 9, then assume that the actual Hamming distance $\mathrm{HD}_{Real} = 23$ and XOR the corresponding precomputed input values with the plaintexts to derive other possible keys. Proceed in a similar fashion if HD_{obs} is even. This attack model is related to Figure 1.9 and Table 2.1.
- If no key candidate stands out from others, repeat the acquisition and analysis phases for a different test input, which in turn leads to another mask value.

An important point before moving to the results is that the attack success probability not only depends on the number of active slices, but also on the number of active scan chains. This is because a smaller number of scan chains can be covered more frequently since a static mask with a lower Hamming weight would be sufficient to include all of them. For instance, let there be c active scan chains in the design. Statistically, once in 2^c different masks, all KFFs will be present and affect the test outputs, therefore giving a better chance of mounting a successful attack. Hence, the smaller number of active scan chains present in the design, the better the success rates will be for the attack. Also it should be noted that even when the number of active scan chains and active slices are fixed, the attack success will be affected by the distribution of KFFs over the scan structure.

Table 2.4 shows DSA success rates in percentages for varying active scan chains and active slices. The results illustrate that there is a substantial fall in the success rates with decreasing number of active slices when the number of active scan chains is kept fixed. Larger number of active slices means a greater spread of KFFs over the scan structure. However, when the number of active slices is fixed, there is a small increase in success rates as the number of active scan chains is reduced. This shows that the dependency of DSA is more on active slices than on active scan chains. This observed behavior is due to KFFs on the same slice being XORed together and therefore reducing the observed Hamming distance value in the test output.

Table 2.4: DSA Success rates for static masking for different distributions

#Active Scan Chains = 32		**#Active Slices = 32**	
#Active Slices	**Success Rate**	**#Active Scan Chains**	**Success Rate**
32	81.91%	32	81.91%
31	77.48%	31	83.26%
30	72.02%	30	83.76%
29	66.79%	29	86.07%
28	63.21%	28	86.14%
27	56.88%	27	88.19%
26	53.24%	26	88.03%
25	49.21%	25	88.30%
24	44.39%	24	89.82%
23	41.25%	23	91.09%
22	37.81%	22	91.77%
21	33.19%	21	92.28%
20	30.39%	20	92.60%
19	27.94%	19	93.13%
18	25.29%	18	93.78%
17	22.63%	17	94.49%
16	20.75%	16	94.22%

Figure 2.4 shows the change in the success rate of the attack with different distributions having the same number of active scan chains and active slices. Two important observations can be made from Figure 2.4. Firstly, the distribution of KFFs over active slices will affect the success rate as it determines how fast the information is processed by the compactor. For instance, when the distribution 22 in Table 2.2 is considered for the active slices, it is clear that nine KFFs will be processed in one test clock. However, for distribution 4, it would take three test clocks to process the same amount of information, which means less information is lost and eventually the observed Hamming distance after the compactor is more likely to enable the attacker to mount a successful attack. Hence, it is realistic to expect a change in success rates which is inversely proportional to the increasing number of KFFs on a single slice. Similarly, when nine KFFs are grouped on a single scan chain, information on each of those KFFs will be included in the output with probability $\frac{1}{2}$ (since these KFFs are XORed together, and can either appear at the output or be masked by other KFFs on the same active slice). If we again compare it to distribution 4 for the active scan chains, the same KFFs would be included in the output with probability $\frac{1}{8}$. Therefore, the wider the distribution

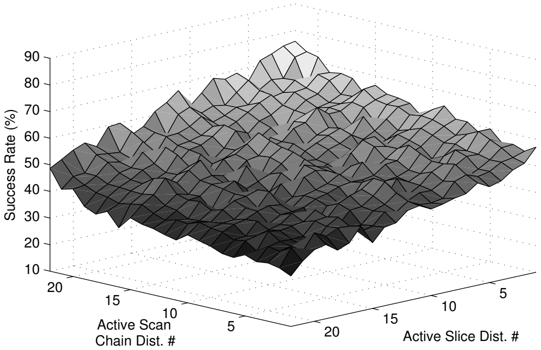


Figure 2.4: Success rate of the attack on OPMISR (space compaction only) for 24 active scan chains and 24 active slices (distributions from Table 2.2)

of KFFs over the active scan chains, the lower success rates one will get when mounting the attack.

In Table 2.2, the lower distribution numbers have a greater spread of KFFs on the scan structure than the higher distribution numbers. As the spread of the KFFs on active slices increases, the attack success rate goes up. This is because the probability of the KFFs on the same active slice getting masked by XOR compaction decreases. Similarly, as the number of active scan chains increases, there is less probability that a larger number of KFFs on the same active scan chain will be masked due to static X-Masking, which leads to higher attack success rates. This gives rise to the structure of Figure 2.4.

Although the above arguments can provide a designer with options towards designing more secure scan chain structures with the same tools that he is using, Figure 2.3 should always be kept in mind when claiming security. It should be noted that, it is possible to make the attacker's job harder, but it is only a matter of resources for the attacker to recover the key when masking schemes are considered.

Applying the attack once in software with 1000 inputs on the design, with a random distribution of KFFs, takes around 0.96 seconds using the configuration mentioned in Section 2.2.1.

2.2.4.3 DSA on XOR Compaction, Static X-Masking and OPMISR

For the sake of completeness, the same attack is applied for a design which uses both static X-Masking and MISRs, therefore having both space and time compaction at the same time. We take a time compaction ratio of 4 as an example. When the attack is applied using the distribution, on which the attack described in Section 2.2.4.1 is most successful, the success rate is reduced to 3.55% from 94.22%. This distribution is in fact the one with 32 active slices and 16 active scan chains which should theoretically provide the highest success rate. Although this gives an idea about the security of including both time and space compaction at the same time in a scan design, it should be noted that the attack here does not include any method to exploit the MISR structure.

Though the attack in [37] can successfully recover the AES secret key in the presence of MISR-based time compaction, it relies on the assumption that the MISR register is observable after each scan clock. In a real-life case, it may be possible to make the parallel outputs of a MISR visible during testing, however it would raise two major issues. Firstly, the gain in using MISRs after the scan structure diminishes as they are supposed to compress the golden value to make the testing procedure more efficient. Secondly, if the MISR content is available at all times, then this implies that the complete scan chain contents are available to the attacker. Therefore using the method proposed in [134] would suffice to recover the key. In this work, we focus only on the output signatures of MISRs, which we believe to be a more realistic assumption, and observe that attacking such a system would not be possible by using simple DSA techniques.

2.2.5 Scan Attack on AES Hardware in the Presence of XOR Compaction with Dynamic X-Masking

In [90], it is claimed that Mentor Graphics' Tessent TestKompress employing Embedded Deterministic Test (EDT) provides inherent security as the scan inputs and outputs are compressed and rendered useless for an attacker. It is stated in [90] that EDT can provide the benefits of deterministic scan test while meeting the security requirements of secure applications. Since its introduction in 2001, Tessent TestKompress, which implements EDT, has been employed in a variety of applications such as smart cards and systems in the defense industry. It is further stated in [90] that the combination of an input decompressor and an output XOR compactor in EDT make it impossible to over-shift the scan chains to understand the scan configuration of the design. Hence, it is further claimed that for most secure applications, this provides a high level of security that meets the intended requirements. In extreme cases, it is possible to customize the EDT hardware to change the order in which scan chains are connected to the compression logic. This further camouflages the design and its scan chains from anyone with intimate knowledge of Tessent TestKompress and its default settings.

A theoretical security analysis of EDT is provided in [81]. In this paper, it is claimed that scan attacks on designs with embedded decompressor and compactor is extremely difficult and these compression structures can achieve a high security level. However, the security claim in that work is based on the assumption that a scan attack requires knowledge of the internal test structure and secret registers, which may not always be valid.

In dynamic masking, as employed in EDT, parts of the scan chains are discarded by a dynamically generated mask updated at each clock or after some number of clock cycles. There may be an optional decoder clock to set the mask frequency matching with the test clock or a separate mask clock. For the sake of simplicity, let us assume that the mask is updated at each clock cycle. In this case, the attack complexity will depend on the number of KFFs in the design, since at each clock the probability for a particular KFF to be contributing to the XOR compactor is $\frac{1}{2}$. Hence, even if there are multiple KFFs on the same scan chain, they have to be unmasked at the appropriate clock cycle which leads to having an attack complexity independent of the distributions of KFFs in the scan design. Therefore, ideally, a successful attack is expected to require $2^{(\#\text{ of KFFs})}$ which would be 2^{32} for AES. On the other hand, if the mask is changing only after some clock cycles and if this number is fixed for the design, the attack complexity will definitely be affected in the favor of the attacker.

Similar to the attack described in Section 2.2.4, the scan attack principle remains the same. However, the only difference is that the mask used in the emulation of the attack is dynamically changing depending on the test input. First, all 256 possible values are given to the first byte of the plaintext and the first round outputs are collected. Then, the outputs are paired depending on the selected input XOR difference and a key guess is made. The attack is again repeated for a number of times to be able to distinguish the correct key candidate from others.

Table 2.5 shows DSA success rates in percentages for varying active scan chains and active slices. Similar to the static X-Masking case presented in the previous section, the results illustrate that there is a substantial fall in the success rates with decreasing number of active slices when the number of active scan chains is kept fixed. However, when the number of active slices is fixed, the success rate stays almost the same. This is mainly because the mask is assumed to be updated at each clock, and therefore the probability of all KFFs influencing the output only depends on the total number of KFFs, as each KFF in a different slice should be selected at each clock individually. This shows that the dependency of DSA for dynamic X-Masking is much more pronounced for varying active slices, and there is hardly any dependency on active scan chains. This can act as a guideline for the security design engineer to place the KFFs on specific slices during DfT insertion.

From Figure 2.5, it can be inferred that the argument in the previous section regarding the distribution of KFFs over active slices is also valid for the case of dynamic masking. However, the KFF distribution over the active scan chains will have no effect since the mask is assumed to be updated at each test clock.

Table 2.5: DSA Success rates for dynamic X-Masking for different distributions

#Active Scan Chains = 32		**#Active Slices = 32**	
#Active Slices	**Success Rate**	**#Active Scan Chains**	**Success Rate**
32	82.18%	32	82.18%
31	77.00%	31	81.87%
30	71.73%	30	81.28%
29	66.66%	29	81.37%
28	62.37%	28	82.52%
27	57.49%	27	81.77%
26	54.10%	26	81.60%
25	49.99%	25	81.74%
24	44.14%	24	81.75%
23	40.00%	23	81.48%
22	37.62%	22	81.45%
21	32.49%	21	82.58%
20	29.88%	20	81.64%
19	27.97%	19	81.19%
18	24.78%	18	80.85%
17	22.19%	17	81.88%
16	20.53%	16	81.90%

Therefore, even if the KFFs are grouped in the same active scan chain, it is equally difficult for all of them to be picked as in distribution 1 in Table 2.2. However if the mask update clock is slower than the test clock, then we would see a slight increase in success rate while more and more KFFs are grouped at the same scan chain (as in distribution 22 in Table 2.2). Therefore, the cases presented in this work cover the two extreme possibilities when the mask is not updated and when mask update clock is the same as the test clock, giving an idea of the cases in between.

Applying this attack once in software with 1000 inputs on a design with a random distribution of KFFs takes around 0.88 seconds using the configuration mentioned in Section 2.2.1.

2.3 Differential Scan Attacks on Public Key Implementations

Though most of the work on scan attacks has been targeted towards AES hardware implementations, advances have also been made on demonstrating that public-key

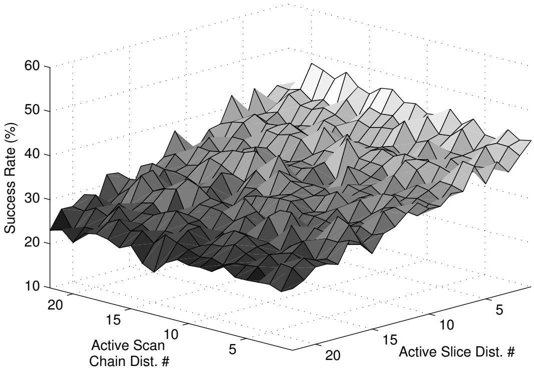


Figure 2.5: Success rate of the attack on EDT for 24 active scan chains and 24 active slices (Distributions from Table 2.2)

hardware implementations are also susceptible to scan attacks.

2.3.1 Differential Scan Attacks on RSA

A new differential scan attack is proposed in this thesis which aims at retrieving the private key from RSA hardware implementations with advanced DfT structures. Moreover, the attack may be applied without knowledge of the DfT structures, which makes it more realistic.

2.3.1.1 RSA

The Rivest-Shamir-Adleman (RSA) algorithm [114] is a widely used public-key cryptographic algorithm, employed in a wide range of key-exchange protocols and digital signature schemes. A brief description of the RSA algorithm is presented in Algorithm 1 and Algorithm 2.

Algorithm 1 RSA key generation

```
1: Select uniformly random primes $p$ and $q$
2: N = $p \cdot q$ (1024 bit)
3: e = random co-prime to $\lambda(N) = lcm(p-1, q-1)$
4: d = $e^{-1}$ mod $\lambda(N)$
```

Algorithm 2 RSA encryption and decryption

```
Ciphertext c = $m^e$ mod N
Decrypted plaintext m = $c^d$ mod N
```

Both the operations in Algorithm 2 are large number modular exponentiations. When RSA is implemented in hardware, there are various options and many algorithms are available. The Montgomery Exponentiation method is most widely used owing to its efficient hardware implementation, as it does away with the expensive division operation required for modular multiplications involved in an exponentiation. Hence we choose the Montgomery method [92] as the target algorithm for our scan chain attack.

The Montgomery product of two n-bit numbers A with B is denoted by: Mont(A, B) = $A \cdot B \cdot R^{-1}$ mod N, where $A \cdot B$ denotes a modular multiplication, N is the modulus or the product of the prime numbers used in the modular multiplications, and R = 2^n. A 1024-bit RSA modulus is used as a case study in this work.

The RSA Montgomery Exponentiation is presented in Algorithm 3 [89]:

Algorithm 3 Montgomery exponentiation

```
INPUT: Modulus $N$, $R = b^\ell$, exponent $e = (e_t ... e_0)_2$ with $e_t = 1$, and an integer $x$,
    $1 \leq x < N$ ($\ell$ is the number of bits in the prime number, 1024 in our case, $b$ is
    the base, which is 2 for binary)
OUTPUT: $x^e$ mod $N$
$\widetilde{x} \Leftarrow$ Mont($x$, $R^2$ mod $N$), $A \Leftarrow R$ mod $N$. ($R$ mod $N$ and $R^2$ mod $N$ may be
    provided as inputs)
for $i$ from $t$ downto 0 do
    $A \Leftarrow$ Mont($A$, $A$)
    if $e_i$==1 then
        $A \Leftarrow$ Mont($A$, $\widetilde{x}$)
    end if
end for
$A \Leftarrow$ Mont($A$, 1)
Return $A$
```

In Algorithm 3 [89], Mont(A, A) is known as the squaring (S) operation, while the Mont(A, $\widetilde{x}$) is known as the Multiplication operation (M). Each iteration of the loop within the algorithm consists either of a squaring and multiply operations if the exponent bit is 1, or only a squaring operation if the exponent bit is 0.

In our proposed scan-based attack, we are focusing on the intermediate register (A, in Algorithm 3) which stores the value after each Montgomery multiplication. Irrespective of how the RSA modular exponentiation is implemented, the

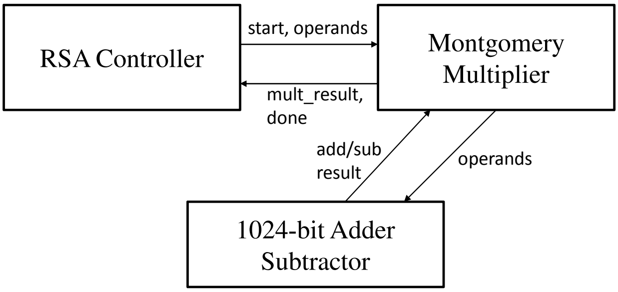


Figure 2.6: Block diagram of the RSA hardware implementation

intermediate value will always be stored in a register. For instance, we may have a hardware/software co-design for the RSA crypto-processor, where the Montgomery multiplier is implemented as a co-processor in hardware (for efficiency) and the control logic or the algorithm for the Montgomery exponentiation implemented in software on a microcontroller. In this case, the results of the intermediate Montgomery operations may be stored in an external RAM, but this value needs to be transferred and stored in the registers inside the Montgomery multiplier datapath to allow the module to perform the computations correctly.

2.3.1.2 Target RSA Hardware Implementation

A hierarchical 1024-bit RSA hardware implementation (employing the Montgomery Exponentiation algorithm) is the target of the proposed scan attack. It consists of an adder/subtractor arithmetic module, a Montgomery multiplier block, and an RSA controller datapath for controlling the square and multiply operations involved in the exponentiation. This is shown in the block diagram in Figure 2.6. The Gezel hardware/software co-design environment [2] was used to create the design; it was transformed into VHDL using the ‘fdlvhd’ VHDL converter tool of Gezel, and finally the Synopsys Design Compiler v2009.06 was used to convert the VHDL file into a gate-level netlist. This implementation does not consider protection against Simple Power Analysis (SPA), Differential Power Analysis (DPA) and Fault Attacks, but test compression techniques supposedly acting as scan-attack countermeasures have been included.

2.3.1.3 Differential scan Attack Mode

One of the main advantages of the attack proposed in this work over the previous RSA attacks is the fact that it works in the presence of industrial DfT structures. For that purpose, the differential mode, as explained in Section 1.3.6, is used to deal with linear response compactors which are inserted by the majority of the DfT tools. Without compaction, the values stored in the SFFs are directly observable at the test output while they are shifted out. On the other hand, in the presence of compaction, each bit at the test output depends on multiple SFFs. In the case of parity compactors, each output bit is the XOR operation between the scan flip-flops on the same slice. It means that the actual value stored in one SFF is not directly observable. Instead, if it differs from the value expected, the parity of the whole slice also differs, and so faults may be detected. This difference is also exploited by an attacker.

Figure 2.7 shows a crypto block and the intermediate register which is the target of the scan attack. The rest of the circuit will be omitted for clarity reasons. The differential mode consists of applying pairs of plaintexts, in this example denoted by (M_0, M_1). The circuit is first reset and the message M_0 is loaded. Then after some fixed clock cycles, the circuit is halted and the intermediate register I_0 is shifted out. The same procedure is repeated for the message M_1 for which I_1 is obtained. Let us suppose that I_0 differs from I_1 in 6 bit positions as shown in Figure 2.7, where a bit flip is represented by a darker box. Let us further suppose that the intermediate register contains only 16 bits and the bits 0, 8, 10, 13, 14, and 15 are flipping. The parity of the differences is equal to 0, since there is an even number of bit flips.

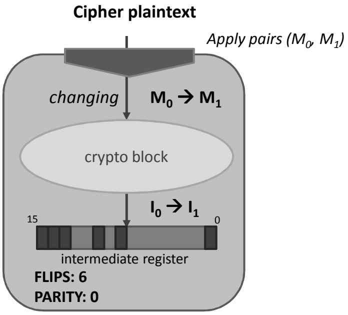


Figure 2.7: Design with crypto block

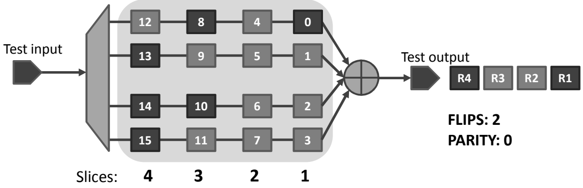


Figure 2.8: Example of DfT scheme

In Figure 2.8, the flip-flops of the intermediary register are inserted as an example of a DfT scenario with response compaction. In this case there are four scan chains divided into four slices. RX represents the test output corresponding to the slice X. As may be seen, if only the bit 0 flips in the first slice (an odd number) this difference is reflected into a flip of R1. In slice 2, no bits flip and thus R2 remains the same. Two flips occur in slice 3: 8 and 10. In this case, both flips mask each other, thus 2 flips (even) result in 0 flips at the output R3. In slice 4, 3 bit flips are sensed as a bit flip in R4.

The parity of flips in the intermediate register is equal to the parity of flips at the output of the response compactor. This property is valid for any possible configuration of the scan chains and slices. Additionally it is also valid for compactors with multiple outputs. In this case, the difference measured should consider all compactor outputs. Thus using the differential mode, the attacker observes differences in the intermediate register and then retrieves the secret key.

2.3.1.4 Description of the Scan Attack on RSA

As presented earlier, the Montgomery exponentiation consists of repeating the Montgomery multiplication operations several times. The first multiplication in the main loop, i.e., the squaring of A, is always performed independent of the value of the secret key bit. The second multiplication, A times $\widetilde{x}$, is performed only if the decryption key bit is 1. The main idea of the attack proposed here is to check if the second operation is executed or not, by observing the value of A afterward. If it does, then the key bit is 1, otherwise it is 0. This procedure is repeated for all the key bits.

In order to detect if the second multiplication was executed, the attacker must scan out the value of A after each loop (timing issues detailed in the Section 2.3.1.5). Additionally, as explained above, a pair of plaintexts is used to overcome the obscurity provided by the response compactor. This pair must be properly chosen so that a difference on the parity of A would lead to the decryption bit. For that, it is important that we give a pair of specific message inputs to the algorithm. The process to derive these ‘good’ pairs of messages is as follows:

First, a pair of random 1024-bit messages is generated using a software pseudo-random number generator (rand() function in C). We denote them here as (M_0, M_1). Then, the corresponding output responses (after one iteration of the exponentiation algorithm) are computed on each of these messages assuming the key bit to be both ‘0’ and ‘1’. Let $(R_{00}, R_{01}, R_{10}, R_{11})$ be the responses for message M_0 and M_1 for key bit ‘0’ and ‘1’ respectively. Let Parity(R_{00}), Parity(R_{01}), Parity(R_{10}) and Parity(R_{11}) be the corresponding parities on these responses. Let P_0 be equal to Parity(R_{00}) XOR Parity(R_{10}) and P_1 be equal to Parity(R_{01}) XOR Parity(R_{11}). If $P_0 \neq P_1$, then the messages are taken to be useful, otherwise they are rejected and the process is repeated till a pair of 'good' messages is obtained. This is illustrated graphically in Figure 2.9.

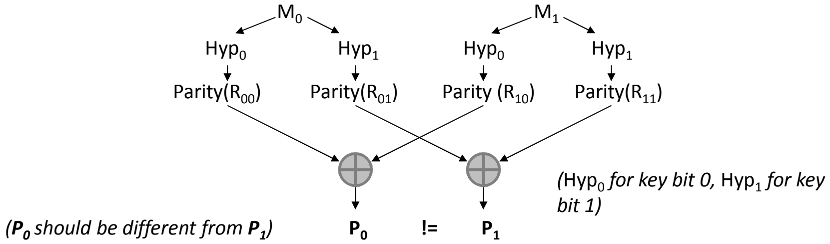


Figure 2.9: Finding good pair of messages for scan attack on RSA

After a good pair of messages is found, it may be applied to the actual circuit. For both pairs of elements, the application is executed in functional mode for the number of clock cycles corresponding to the targeted step (decryption key bit). For these pairs of elements, the scan contents are shifted out and the parity of the difference at the test output bitstream is measured. If the parity of differences is equal to P_0, then the hypothesis 0 is correct and the secret key bit is 0. If it is equal to P_1, then the secret key bit is 1. This procedure is repeated for all the bits of the decryption key.

2.3.1.5 Practical Aspects of the Attack

Performing scan attacks on actual designs requires additional procedures which have not been taken into consideration by some previous attacks proposed in the literature. The two main practical issues consist of (1) dealing with the other flip-flops of the design; (2) finding out the exact time to halt the mission mode execution and to shift out the internal contents. The first issue is solved by analyzing the leakage of the FFs of the intermediate register at the test output as described in Section 2.3.1.5.1. The second issue is described in Section 2.3.1.5.2.

2.3.1.5.1 Leakage Analysis

The scenario of Figure 2.7 and Figure 2.8 is commonly taken into consideration by scan attacks, however in real designs other FFs of the design will be included in the scan chain. These additional FFs may complicate the attack if no workaround is taken into account. Figure 2.10 shows a design containing three types of FF. We define here three types of scan flip-flops (SFFs), depending on the value they store, as shown in Figure 2.11. Class T1 SFFs correspond to the other IPs in the design, that store data not dependent on the secret. T2 SFFs belong to the registers directly related to the intermediate register, that store information related to the secret key and that are usually targeted by attackers (e.g. RSA intermediate register). T3

SFFs store data related to the cipher but not the intermediate registers themselves (such as input/output buffers or other cipher registers). The leakage, if it exists, concerns the T2 type.

The goals of the leakage analysis is to find out if a particular bit of the intermediate register (T2) can be observed at the test output, and locate which output bit is related to it. Let T2N be the value stored in T2 after N clock cycles while the design is running in mission mode from the plaintext M_0 (the first event in mission mode is a reset). The analysis is focused on one bit per time, looking for an eventual bit flip in T2. In order to do that, the pair (M_0, M_1) is chosen so that the value on T2N for M_0 differs by a single bit from the value T2N for M_1. In Figure 2.10 the darker blocks represent a bit that flips. Thus, in this case, the least significant bit of T2N flips. Since the attack tries to verify if it is possible to observe a flip in the LSB of T2N, it is ideal that there is no flip in T1N. To reduce the effect of the T1 flip-flops, all the inputs that are not related to the plaintext are kept constant. It means that T1N for M_0 has the same value as T1N for M_1. However, the same method cannot be applied to reduce the effects of T3. Since we suppose that the logic associated with T3 is unknown and since its inputs are changing, the value T3N for M_0 may differ from T3N for M_1. In our example, let us flip only three bits of T3.

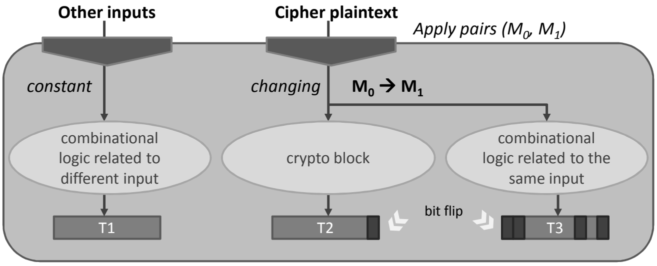


Figure 2.10: Generic Cryptographic Design showing categories of FFs

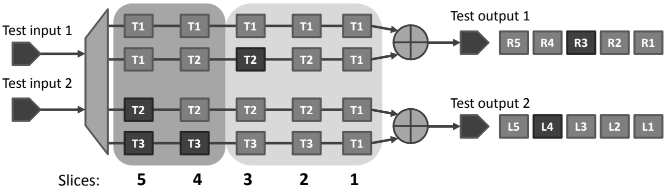


Figure 2.11: Test Compression with multiple scan outputs

Figure 2.11 shows the result of these bit flips in the scan chain and consequently in the test outputs. As an example, we suppose that the DfT insertion created 4 scan chains, and placed a pattern decompressor at the input and a response compactor with two outputs (R and L). As may be seen, slice 1 contains only T1 scan flip-flops, meaning that after the response compactor, the values of R1 and L1 are not supposed to flip (because T1N has the same value for M_0 and M_1). For slice 2, the same happens. Slice 3 contains the only flipping bit of T2N and the other flip-flops in the slice do not change. In this case, the bit flip of the LSbit of T2N is observable in R3. It means that an attacker could exploit the information contained in R3 to find the secret key. Hence, this is considered a security leakage and may be exploited by the attack described in Section 2.3.1.4.

Slice 4 and slice 5 contain flip-flop configurations that may complicate an attack. For instance in slice 4, there are FFs of T1 and T2 that are not affected by a change from M_0 to M_1. However it contains one FF affected in T3. It implies that the value on L4 flips, which may confuse the attacker (he expects a single bit flip caused by the LSB of T2). In this case, the attacker is able to identify that the value of L4 is dependent on the plaintext, but is not able to exploit this information, since T3 related logic is supposed to be unknown. Another complication is shown in the configuration of slice 5. If the LSB of T2 is actually on the same slice as a flipping SFF of T3, the flip is masked and no change is observed in L5. In this case, the attacker is not able to exploit that bit.

Next, the attacker repeats this method for each bit of the intermediary register (e.g., 1024 times for 1024-bit RSA). If some useful leakage (like R3) is detected, he proceeds with the attack method explained in the Section 2.3.1.4.

2.3.1.5.2 Timing Aspects

The scan-based attack on RSA is targeted at finding the decryption key. It is very important to find the exact time to scan out the contents of the intermediate registers using the scan chains. The timing aspects for the attack are presented pictorially in Figure 2.12.

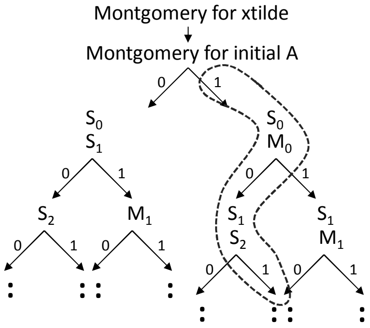


Figure 2.12: Timing estimation tree for scan attack on RSA

Since the same hardware is commonly used for both encryption and decryption in RSA, we can run the hardware with a known encryption key in order to get the timing estimations. For instance, the attacker must find out the number of clock cycles that a Montgomery multiplication operation takes. With a known key, we know the number of Montgomery multiplications required for the square and multiply operations of the RSA modular exponentiation (Algorithm 3). Dividing the total time of execution for this encryption by the number of operations gives the approximate time required for one Montgomery operation. Then using repeated trial-and-error steps of comparing the actual output with the expected result after one Montgomery multiplication, it may be possible to find out the exact number of clock cycles required.

This timing is utilized in our attack during the decryption process to find out the decryption exponent. The RSA in hardware is run in functional mode for the exact number of cycles needed to execute a predetermined number of Montgomery operations. Then the hardware is reset, scan enable is made high and the scan-chain contents are taken out. Depending on whether the key bit was 0 or 1, either a squaring (S) is performed or both square (S) and multiply (M) are performed respectively. In our proposed attack, we always run the software implementation for two Montgomery cycles taking the key bit as 0 and 1 (two hypothesis in parallel). If the first bit was 1, both square (S_0) and multiply (M_0) operations are performed, otherwise two squarings (S_0 and S_1) are performed. Then the actual result from the scan-out of the hardware implementation after each key bit execution is checked with the results of the simulation in software. If it matches with the first result (of S_0 and M_0), then the key bit is 1, otherwise the key bit is 0. Now, for the next step starting with the right key bit, again the decryption is performed in software assuming both 0 and 1 possibilities. This time we run for one or two Montgomery cycles depending on whether the previous key bit was 0 or 1 respectively. If the previous key bit was 0, then squaring on the next key bit (S_2) is performed for key bit 0 and a multiply on the same key bit is performed (M_1) for present key bit 1. On the other hand, if the previous key bit was 1, then squaring on the same (S_1) and next key bit (S_2) is performed for present key bit 0 or a square (S_1) and multiply (M_1) on the same key bit is performed (M_1). The results are compared with the actual result from the scan-out of the hardware implementation, and the corresponding decision is taken. This iterative process is repeated until all the decryption key bits are obtained. As an example, if the decryption key bits were 101..., the timing decision tree would follow the path denoted within the dotted lines in Figure 2.12 (S_0, M_0, S_1, S_2, M_2, ...).

The complexity of the timing analysis can be represented as $O(|e|)$, where $|e|$ represents the exponent length. To reduce the complexity of the attack, backtracking algorithms such as in [75], can be employed.

2.3.1.6 Attack Tool

In order to apply the attack to actual designs, we developed an attack tool. The main goal of this tool is to apply the attack method proposed in Section 2.3.1.4, as well as the leakage analysis proposed in sub-section 2.3.1.5.1, to many different DfT configurations, without modifying the attack principle.

The scan analysis tool is divided into three main parts: the attack methods and ciphers (implemented in C++), the main controller (in Perl), and the simulation part which is composed of an RTL deck and ModelSIM scripts, as may be seen in Figure 2.13. In order to use the tool the gate-level netlist must be provided by correctly setting the path for both the netlist and technology files for ModelSIM simulations. Then the design is linked to the RTL deck, which is used as an interface with the tool logic. This connection is automatically done by giving the list of input and output data test pins, as well as the control clock, reset, and test enable pins. Additionally, other inputs such as plaintext and ciphertext must be set in the configuration file.

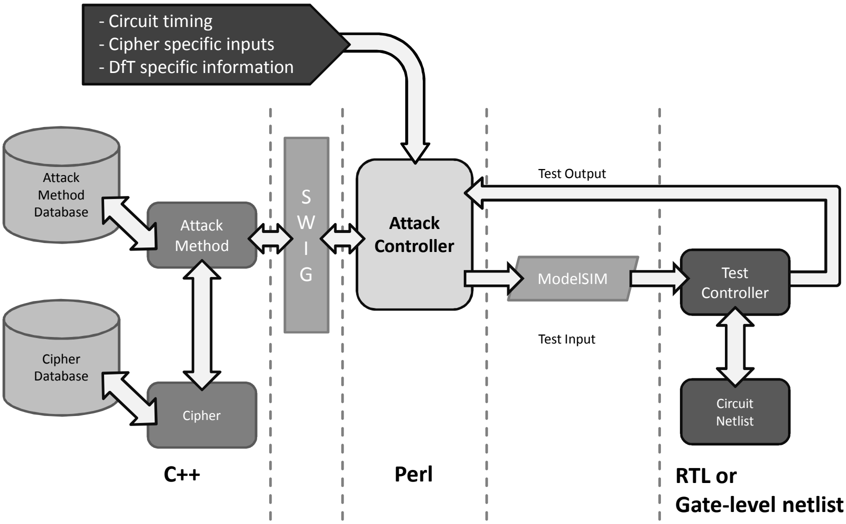


Figure 2.13: Scan attack tool

Once the Device-Under-Test (DUT) is linked, the tool may simulate it by calling ModelSIM SE with the values passed by the main controller. This interface is achieved by setting environment variables in the Perl script which are read by the ModelSIM Tcl script and then passed on to the RTL deck via generics. For instance, the information being exchanged here is the plaintext (cipher specific), reset and scan enable timing (when to scan and for how long) and the value of the scan input (test specific). In return, the scan output contents are stored in a file and they are processed by the main attack controller in order to run the attacks.

On the left side of Figure 2.13, the software part is shown (attack method and cipher description). The new RSA attack method is written in C++ for an RSA software implementation. Scan-attacks on other similar cryptosystems may be conceived since the tool was built in such a way that adding a new cipher is simple.

The core of the tool is implemented by the attack controller (Perl) that calls the attack method through a Simplified Wrapper and Interface Generator (SWIG) interface. SWIG helps connect programs written in C/C++ with other high-level programming languages such as TCL/Perl. The attack controller ensures that the settings are initialized and then it launches both the attack and the simulation. As a secondary functionality, the controller handles some design aspects, like timing and multiple test outputs, so that the attack method itself may abstract that information. For instance, the attack method has no information on how many clock cycles it takes to execute a Montgomery multiplication. Also, it finds out the number of scan cycles that the shift operation must be enabled so that the contents of the scan chains are completely unloaded.

2.3.1.7 Experimental Results

In order to test the effectiveness of the attack, we implemented a 1024-bit RSA algorithm in hardware with separate datapaths for the Montgomery multiplier, adder/subtractor block and the main controller for the Montgomery exponentiation. Then we envisaged different scenarios to test the attack flexibility. The first is a single scan chain containing all the FFs of the design. Secondly, we used Synopsys' DfT Compiler (v2010.12) to insert more complex configurations such as decompression/compaction. Finally, we implemented some countermeasures proposed in the literature to verify if the attack is able to overcome them. All the runs were performed on a 4 GB Intel Xeon CPU X5460 with four processors.

The total number of FFs in the design is 9260. Out of these, 4500 belong to the T1 class, 1024 consist of the intermediate register (T2 class) and 4096 belong to the T3 class (see Section 2.3.1.5.1). Using Synopsys' DfT Compiler, we inserted a single chain with all these FFs, and the design was linked with the tool. Then the leakage analysis was run over this configuration. The tool proceeds with the attack method in order to find the secret key. In this phase, it takes again approximately 3.5 min to recover each bit of secret key. Both the timing for the leakage analysis and the attack are strongly dependent on the server configuration. Additionally, the C++ code takes approximately 5 seconds from the 3.5 minutes, meaning that the simulation in ModelSIM limits the execution time.

For our test case, we required around 11 messages to find out the full 1024-bit RSA exponent. This number is less than that required for the attack presented in [96] (which takes around 30 messages).

Scan attack on RSA in the presence of advanced DfT Methods: In order to test our scan attack in the presence of industrial DfT methods, Synopsys' DfT Compiler was used to insert different DfT configurations in the RSA circuit. In the first case, 120 scan chains were inserted, without compaction/compression. Since the tool analyzes each scan output pin separately and independently, and since the sensitive registers were converted to scan FFs, the attack with the tool was able to find out the secret key. Changing the position of the sensitive FFs does not change the result. The time taken to retrieve the key in this case is almost the same as that of the previous case (with a single chain).

In a second scenario, response compaction (explained in Section 1.2.1) was inserted. Since the test inputs are not used in the response compactor (the plaintext is a primary input), it does not affect the attack method and hence it is not taken into consideration. As the proposed methods are all based on the differential mode, the linear XOR response compactors do not impede the attack and also it does not imply a significant increase in simulation time.

As a last scenario, X-Tolerant structures (explained in Section 1.2.2.1) were activated to add the masking logic that deals with the unknowns (X-states) present in the design. The masking blocks some scan chains at the instant while the contents are shifted out if the test engineer believes that there is an X-state that may corrupt the test output. This mask is controlled by the output of the pattern decompressor, which is in turn controlled by the test inputs. Since the mask is controllable, it is just a matter of shifting in the right pattern which does not mask the confidential data. Thus the masking can set when the sensitive data is shifted out. Hence, our proposed scan-attack still works in the presence of masking.

2.3.1.8 Previous Work

RSA private keys have been retrieved through scan attacks by methods described in [96]. The attack method is based on observing the values of the intermediate register (the register storing the results of the intermediate square and multiply operations) on the scan chain for each bit of the secret key (decryption exponent for RSA), and then correlating this value with a previous offline calculation, which the authors refer to as 'discriminator'. If the value matches with this discriminator value, a corresponding decision is taken on the key bit.

In [96], a pure software attack is proposed which does not take into account the practical aspects of applying it to an actual cryptographic hardware implementation. The timing aspects are crucial to scan attacks on secure hardware, which has been addressed in this work. Our scan-attack analysis tool integrates the actual hardware (in the form of a gate-level netlist with inserted DfT) with the software emulation which allows us to perform the attack in real-time. The secret decryption exponent key bits are recovered on-the-fly using this combined approach.

Left-to-right binary exponentiation (employed in ordinary exponentiation) is used as the target RSA algorithm for the attack in [96]. This is generally not implemented in hardware owing to the expensive division operation involved in modular operations. We target the Montgomery Exponentiation algorithm, which is by far the most popular and efficient implementation of RSA in hardware, as there are no division operations involved (owing to performing the squaring and multiplication operations in the Montgomery domain).

Moreover, an inherent assumption of the attack in [96] is that there are no other exponent key-bit dependent intermediate registers which change their value after each square and multiply operation. This may not be the case in an actual hardware implementation, where multiple registers are key dependent and change their values together with the intermediate register of interest in the attack (for instance, input and output buffers). These registers may mask the contents of the target intermediate register after XOR-tree compaction (as shown in the leakage analysis in Section 2.3.1.5.1). Our proposed scan-attack analysis takes the contents of other key-dependent registers present in the scan chain, and presents ways to deal with this problem.

Finally, the attack in [96] cannot be applied to secure designs having test response compaction and masking (which is usually employed in DfT for all industrial circuits to reduce the test volume and cost). Our scan-attack analysis, on the other hand, works in the presence of these scan compression DfT structures.

2.3.2 Differential Scan Attacks on ECC

2.3.2.1 Elliptic Curve Cryptosystems

For public-key cryptographic implementations, Elliptic Curve Cryptography (ECC) [74, 91] provides equivalent security to RSA with much smaller key sizes (160 bit ECC is equivalent to 1024-bit RSA). An elliptic curve E over a finite field F_p for a prime p may be defined as the set of points P(x, y) satisfying an elliptic curve equation of the form:
$y^2 \ mod\, p = x^3 + ax + b \ mod\, p$, where x, y, a and b belong to a finite field F_p. In this work, we are using the 192-bit NIST ECC prime field curve; the curve parameters used in our ECC implementation are obtained from [59].

The scalar or point multiplication of P with k is generally performed using the Montgomery Powering Ladder [93] (shown in Algorithm 4). This is one of the most efficient methods of performing scalar multiplication, as it does not require

Algorithm 4 Montgomery Ladder for Elliptic Curve Cryptography

```
INPUT: Scalar $k = (1, k_{m-2}, ..., k_1, k_0)_2$, base point P $\in$ E($F_p$)
OUTPUT: $Q_0$ = k.P
$Q_0 \Leftarrow$ P
$Q_1 \Leftarrow$ 2P
for $i$ from $m - 2$ downto 0 do
    if $k_i$==0 then
        $Q_1 = Q_0 + Q_1$, $Q_0 = 2\ Q_0$
    else
        $Q_0 = Q_0 + Q_1$, $Q_1 = 2\ Q_1$
    end if
end for
Return $Q_0$
```

any extra storage and has fast calculation times compared to other methods, thus allowing its efficient implementation. Moreover in this algorithm a point addition and doubling operation is performed in each iteration of the main loop, making it resistant to simple power analysis attacks.

2.3.2.2 Attacker Scenario

The attack proposed in this work can be classified as a known-point attack, where the attacker knows the point P, but does not necessarily control it. We consider the case that the attacker can observe the result of each ECC point multiplication with the help of the DfT structure. The base point is assumed to be stored in non-volatile memory (ROM) while the scalar for the point multiplication is stored in RAM, as it keeps on changing. This is the usual case with most smart-card implementations.

2.3.2.3 Target ECC Implementation

As seen in Algorithm 4, a point multiplication is performed through a series of point addition ($Q_0 + Q_1$) and point doubling (2 Q_0 or 2 Q_1) steps. We perform both point addition and point doubling in projective coordinates for avoiding the costly modular inversion operation. In projective coordinates, Q_0 and Q_1 cost each three 192-bit registers (Q_0[X], Q_0[Y] and Q_0[Z] and so on). There are various methods of performing point addition and doubling. We have used the 1998 Cohen-Miyaji-Ono mixed coordinates [30] for point addition, and the 2007 Bernstein-Lange formulae [31] for point doubling from the Explicit Formulae database [1].

Irrespective of the implementation method, intermediate values Q_0 and Q_1 are temporary stored in Secret flip-flops (SFFs, as defined in Chapter 1) to be available at the next iteration step.

2.3.2.4 Differential Scan Attack on ECC

Contrary to the previous ECC scan attack [94], our approach works in the presence of test compression and X-Masking. Similar to the scan attack on RSA presented in Section 2.3.1.3, we use the differential mode [134, 35] to overcome the obscurity caused by response compactors inserted by most of the industrial DfT tools.

Figure 2.14 shows a small example of a crypto core and the intermediate register that is inserted in a multiple scan-chain circuit with response compaction. The rest of the circuit is not shown to give a more clear description. Real scenarios where other circuit registers are present are discussed later. Boxes S0 to S11 represent the 12 SFFs in the intermediate register. Due to response compaction, the test response R1 stores the parity of S0 to S3 (slice 1), R2 stores the parity of S4 to S7 (slice 2) and R3 stores the parity of S8 to S11 (slice 3). It means that the actual value stored in any of these SFFs is not directly observable at the test responses. However, a difference in one of these SFFs implies a difference in the slice parity and then at the response. This can be exploited by the attacker.

In the differential mode, a pair of plaintexts (points in the case of ECC) is applied, for example (P_0, P_1). The circuit is first reset and the message P_0 is loaded. Then after N clock cycles of mission mode (N depending on the cryptographic implementation), the circuit is halted and the intermediate register state I_0 is shifted out. The same procedure is repeated for the message P_1 for which I_1 is obtained.

To illustrate this, let the Hamming distance between I_0 and I_1 be equal to 6, as shown in Figure 2.14. In this example highlighted boxes represent a bit flip between I_0 and I_1. We also suppose that the bits S0, S5, S6, S7, S8 and S10 are flipping. The parity of the differences at the intermediate register is equal to 0, since the Hamming distance is even. The flip at S0 is sensed at the test response R1. R2 flips due to an odd flips at slice 2, while R3 does not flip since slice 3 has an even number of flips.

2.3.2.5 Proposed Distinguishing Scan Attack

In this section, we show how the parity may be used as a distinguisher in order to retrieve the secret key using a chosen point attack. We first consider that only the SFFs are inserted in the scan chain. Later, the practical aspects of leakage analysis are shown, when there are other FFs changing with the SFFs in the scan chains.

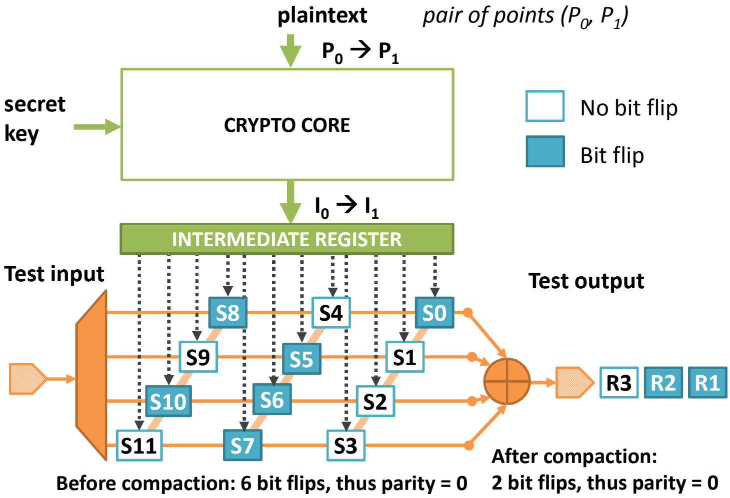


Figure 2.14: Hamming Differences in the intermediate register observable at the test responses

The Montgomery Powering Ladder method consists of repeating point addition and point doubling operations. For each iteration, the value stored in Q_0 is the result of the point doubling if the bit of key is 0; otherwise Q_0 stores the result of the point addition. Thus observing Q_0 allows the detection of the current value of the key bit. This procedure must be repeated for the entire key length (192 bits in our example).

In order to detect if Q_0 stores the value of the point addition or the value of the point doubling, Q_0 must be scanned out. Differential attacks use pairs of plaintexts to overcome the obscurity due to response compaction. In the case of ECC, plaintexts are actually the input points. This pair must be properly chosen so that a difference on the parity of Q_0 would lead to the key bit. The process to derive 'good' pairs of points is as follows (see Figure 2.15):

These useful pairs are software generated. First, a random pair of 192-bit points is obtained using a software pseudo-random number generator. We denote them here as (P_0, P_1). Then, the corresponding output responses (after one iteration of the multiplication algorithm) are computed for each of these points assuming the current key bit to be '0' (Hypothesis Hyp_0) or '1' (Hypothesis Hyp_1). Let R_{ij} be the response to point P_i with a current key bit j. For instance, (R_{00}, R_{01}, R_{10}, R_{11}) are the responses to points P_0 and P_1 for key bit '0' and '1' respectively. Let Parity(R_{00}), Parity(R_{01}), Parity(R_{10}) and Parity(R_{11}) be the parities of these responses.

Let D_0 be the Hamming distance between Parity(R_{00}) and Parity(R_{10}) and D_1 be

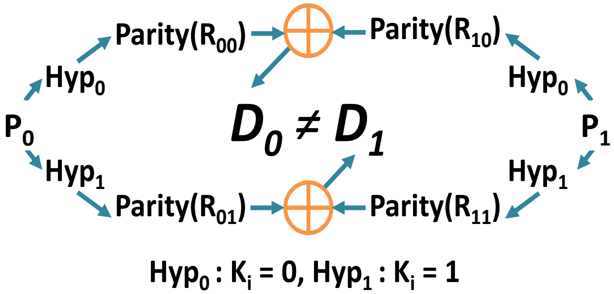


Figure 2.15: Hypothesis Decision for scan attack on ECC

the Hamming distance between Parity(R_{01}) and Parity(R_{11}). If $D_0 \neq D_1$, then the points (P_0, P_1) are taken to be useful, otherwise they are rejected because the value of the current key bit cannot be deduced from the responses to (P_0, P_1). This process is thus repeated till a pair of 'good' points is obtained. For any observable subset of SFFs, choosing a random pair will have 50% probability of being a good pair of messages. Therefore, even if the first pair is not a good pair, choosing 6 random pairs, each one of them will be a good pair of messages with a probability of 99% (1 - $1/2^6$). In practice, we find the pairs very quickly (in the first, second or third pair).

After a good pair of points is calculated, it is applied to the actual circuit. For both elements of the pair, the application is executed in mission mode for the number of clock cycles corresponding to the targeted step (key bit). Then, switching to test mode, the scan contents are shifted out and the difference of parities at the test output bitstream is measured. If it is equal to D_0, then the hypothesis 0 is correct and the secret key bit is 0. If it is equal to D_1, then the secret key bit is 1. This procedure is repeated for all the bits of the secret scalar.

Figure 2.16 depicts an example on the choice of a good pair of points. The points P_0 to P_7 represent a set of points which contains at least one pair that verifies the good pair property for each bit of the secret key. I_0 to I_6 represents the intermediate register Q_0 (or Q_1 depending on the attack target). As can be seen, for retrieving the first bit of the key, the attacker needs a single pair of points (P_1, P_6). Then for retrieving the second bit of the secret scalar, if the pair (P_1, P_6) does not satisfy the good property, then a third point must be added to the set of points, that satisfies the good property when combined with at least one of the previous pairs (P_1 or P_6). In this case P_4 verifies the good property with P_6. Verifying the good property of a new point against all the previous points reduces the number of required points progressively. Around 6 points on average are required to be tried for the first good pair and reduces for each subsequent good pair to around 1-2 points for the last one. If we do not reuse the pairs, we would need one pair for each bit of the private key, therefore 192 for ECC with 192 bits.

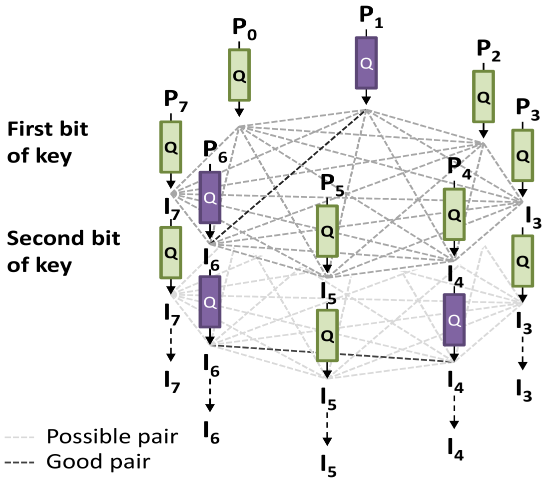


Figure 2.16: Finding a good pair of points for ECC

2.3.2.6 Leakage Analysis

In the procedure above we assumed that the entire state of Q_0 is inserted at the scan chain (Q_1 can be targeted as well without any additional issue) and that there are no other flip-flops in the scan chain. However, in presence of partial scan or X-Masking logic, which is used to prevent unpredictable internal states that can corrupt the test output, part of the Q_0 register may be filtered and thus not totally observable. In addition, scan chains may include scan FFs not related to the crypto core. We detail here the proposed attack in such cases.

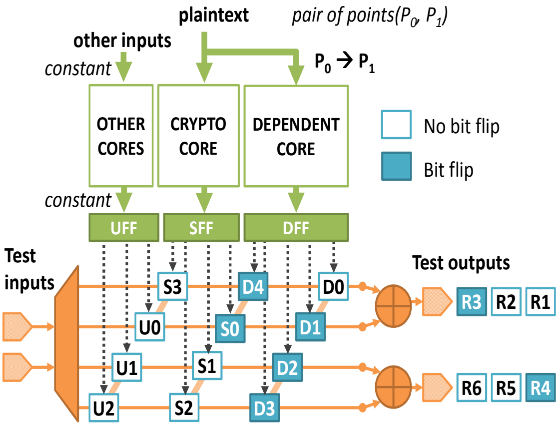


Figure 2.17: Leakage Analysis for ECC

Figure 2.17 shows a small illustrative design containing three types of FFs, depending on the value they store. UFFs correspond to the other cores in the design that store data unrelated to the secret. SFFs belong to the registers directly related to the intermediate register, which store information related to the secret key. DFFs store data related to the cipher but not the intermediate registers themselves (such as input/output buffers or other cipher registers). The leakage, if it exists, concerns the SFFs. Figure 2.17 shows examples of different scenarios of test responses that are classified in three main cases:

- Unrelated Responses: Unrelated responses are the ones that are not related to the secret. In other words, the response depends only on UFFs (it is the case of R6).
- Exploitable Responses: Exploitable responses are the ones that depend only on SFFs or on SFFs plus UFFs (like R3 and R5). The attacker can observe SFF flips at these response bits. It must be noticed that UFFs cause no difference since the other inputs are kept constant during the differential procedure.
- Non-exploitable Responses: Non-exploitable responses are the ones that depend on at least one DFF (like R2, R4 and R1). The attacker cannot observe SFF flips due to the presence of DFFs that are affected by the plaintext input.

The goal of the leakage analysis is to find out if a particular bit of the intermediate register (SFF) can be observed at one of the test response bits. Thus the analysis is focused on a single SFF at a time, looking for an eventual bit flip. In our example, we start with S0. We denote $S0_i^N$ as the value stored in S0 after N clock cycles while the design is running in mission mode from the point P_i (the first event in mission mode is a reset). Similar definitions hold for the other scan FFs. In order to find out which test response bit is related to S0, we use the differential procedure described earlier with a special set of pairs $(P_i, P_j)_0$ with the following property:

$(P_i, P_j)_0 \, : \, S0_i^N \neq S0_j^N$, and $SX_i^N = SX_j^N$ for X = 1 to 3.

In other words the pairs $(P_i, P_j)_0$ cause a single bit flip on the SFFs (in this case: S0), as highlighted in Figure 2.17. Unfortunately, S0 is compressed with D4 that may eventually flip for the pair $(P_i, P_j)_0$ and thus mask the S0 flip. This is a case where R3 is not exploitable and thus differences on S0 are not observable.

If we repeat the same procedure for S1, S2 and S3, with respective set of pairs $(P_i, P_j)_1$ $(P_i, P_j)_2$ $(P_i, P_j)_3$, we will notice that the only test response bits that always change are R3 and R4 (they are not masked by any DFF).

In the case of ECC, this method must be repeated for all the intermediate register bits (192 times), until one leakage is found. It must be noticed that all the 192 bits of the intermediate register depend on the secret scalar. Thus observing at least one of them through the test response allows the attacker to retrieve the entire secret scalar (using the good pairs described earlier). The last 64 bits can also be found out using the Pollard ρ method [106].

Finding set of pairs $(P_i, P_j)_X$ is rather easy since the first two operations to be executed in Algorithm 4 are independent of the key, but the results are stored in the intermediate register.

2.3.2.7 Inherent Countermeasure

It must be noticed that the third case described in the previous section may be considered as an inherent countermeasure against scan-attacks: if each SFF is in the same slice as a DFF. However, this solution contains two drawbacks. The first one is that it is not easily implementable because the position of the scan flip-flops is set by place and route (P&R) tools, and choosing the position will require internal modification of that design step. The second and most important issue concerns the security of this solution. Here we supposed that the attacker does not know the relationship between DFFs and the plaintext, in order to propose an attack that does not depend on deep knowledge of design details. There may be attackers who know this kind of information, which would compromise this countermeasure. This is a case of security by obscurity.

2.3.2.8 Timing Estimate

The scan-based attack on ECC is targeted at finding the secret scalar (192 bits in our case). It is crucial to find the exact time to scan out the contents of the intermediate registers using the scan chains.

Since the same hardware is commonly used for both encryption and decryption, we can run a second hardware instance with a known scalar in order to get the timing estimations. For instance, the attacker must find out the number of clock cycles that a pair of point operations takes. For our target Montgomery Powering Ladder, it is the sequence of point addition and doubling operations. With a known scalar, we know the number of point operations required for the addition and doubling operations of the ECC point multiplication (Algorithm 4). Dividing the total time of execution for this multiplication by the number of pairs of operations gives the approximate time required for one combined point addition and doubling operation. Then using repeated trial-and-error steps of comparing the actual output with the expected result after one pair of point operations, it may be possible to find out the exact number of clock cycles required.

This timing is utilized in our attack during the decryption process to find out the secret scalar. The ECC module is run in functional mode for the exact number of cycles needed to execute a predetermined number of point operations. Then the hardware is reset, scan enable is made high and the scan-chain contents are taken out. Irrespective of whether the secret scalar key bit was 0 or 1, a point addition and doubling operation is always performed. In our proposed attack, we always run the software implementation for a pair of point operations. Then the actual result from the scan-out of the hardware implementation after each key bit execution is checked with the results of the simulation in software. If it matches with the result of a point addition followed by point doubling for Q_1, then the key bit is 0, otherwise the key bit is 1. Similarly if the hardware simulation result matches with the software result of a point doubling followed by point addition for Q_0, then the key bit is 0, otherwise the key bit is 1. This process is repeated for all the secret scalar bits.

2.3.2.9 Scan Attack Experimental Setup

In order to perform the attack in actual designs, the scan attack tool presented in Section 2.3.1.6 is employed. As mentioned earlier, it has the following features: ability to find the possible leakage points, possibility to automatically apply the attack method to a given gate-level design, and applicability to different DfT configurations.

2.3.2.10 Experimental Results and Discussion

In order to test the effectiveness of the attack, we implemented a 192-bit ECC algorithm in hardware using the Gezel environment [2], and then synthesized the gate-level design using Synopsys' DfT Compiler (v2010.12) with a TSMC 130nm library. The total number of FFs in the design is 3355. Out of these, 855 belong to the UFF type. Q_0 consists of 576 FFs, from which each projective coordinate (Q_0[X], Q_0[Y] and Q_0[Z]) has 192 FFs (SFF type). Q_1 contains 576 FFs as well (also SFF type). And finally 1348 belong to the DFF type. For all the test cases, Synopsys' DfT Compiler was used to insert the DfT structures, including decompression/compaction and X-Masking logic. All the runs were performed on a 4 GB Intel Xeon CPU X5460 with four processors.

2.3.2.10.1 Scan attack timing and number of points

Concerning the attack timing, it consists of two phases: identification of leakage points, as described earlier; and the use of these observable points to perform the attack. The first phase takes approximately 49.37 seconds per bit (in average) and the second one takes 49 seconds. 89% of both the timings are due to the

Table 2.6: DfT Configurations

DfT Configuration	# of scan chains	Compression rate	X-Masking	# of points	Attack successful
Config. 1	1	No compression	No	8	Yes
Config. 2	120	No compression	No	8	Yes
Config. 3	120	12	No	8	Yes
Config. 4	120	24	No	8	Yes
Config. 5	120	12	Yes	9	Yes

design simulation in ModelSIM SE, the remaining time comes from the C++ code execution.

For our test case, we required around 8 points over the elliptic curve to find out the secret multiplier k. In other words, all the necessary good pairs (P_0, P_1) are created from these 8 points. This number is less than that required for the attack presented in [94], which takes around 29 points.

2.3.2.10.2 Scan Attack on ECC in Presence of Advanced DfT Methods

A set of representative DfT configurations for which the scan attack was proven to be effective are shown in Table 2.6. Config. 1 and Config. 2 have no compression structures. Config. 3 and Config. 4 have 120 scan chains having compression rates equal to 12 and 24 respectively. Config. 5 shows a test case where X-Masking is activated. In Table 2.6, ‘attack successful’ means that all the bits of the secret scalar were retrieved.

2.3.2.11 Previous Work

The scan attack on ECC in [94] is directed at the Montgomery multiplication method. The attack principle is based on observing the values of the intermediate register of interest on the scan chain against each bit of the secret (scalar multiplier), and then comparing this value with a pre-computed offline value, termed as a ‘discriminator’ by the authors.

However, the attack presented in [94] does not take into account the effect of other secret dependent FFs on the scan chain that may corrupt or mask the scan signature. In the proposed attack presented in this thesis, a detailed leakage analysis is performed and the attack works in differential mode. This scan attack is successful even if there are other FFs that change their values along with the intermediate register of interest. Fewer messages are required to find the ECC secret key compared to the earlier work. The attack in [94] requires 29 points on an average and 35 points in the worst case to find a 163-bit ECC secret scalar. The attack presented here, on the other hand, requires 8 points on an average to retrieve a 192-bit secret scalar.

2.4 Conclusion

This chapter evaluates the security provided by industrial test compression schemes against differential scan attacks on cryptographic hardware implementations. Scan attack results for all the three major DfT tools from Synopsys, Cadence and Mentor Graphics are presented for the block cipher AES. It is demonstrated that for AES, space compression with X-handling logic is vulnerable to scan attacks, whereas MISR-based time compaction can protect against scan attacks provided only the final compacted signature is observable. For the public-key hardware implementations of RSA and ECC, timing and leakage analysis are performed for studying the practical aspects of the scan attack. Requirements of the number of input messages/points for retrieving a complete private key/secret scalar are also presented for different DfT configurations. Eleven messages on average are required for successfully retrieving the private key from a RSA hardware implementation using scan attacks, while a maximum of nine points is required to retrieve the secret scalar from a ECC hardware implementation in the presence of test compression and X-Masking. For public-key based cryptographic hardware implementations, all the test compression schemes with X-Handling logic are found to be insecure.

Chapter 3

Scan Attack Countermeasures

Publication Data

This chapter is based on the following publications:

[A] J. Da Rolt, A. Das, S. Ghosh, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “Scan Attacks on Side-Channel and Fault Attack Resistant Public-Key Implementations,” Journal of Cryptographic Engineering (JCEN), Springer-Verlag Berlin Heidelberg, vol. 2, no. 4, pp. 207-219, 2012.

[B] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, “Security Analysis of Industrial Test Compression Schemes,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2013, vol. 32, no. 12, 2013.

[C] B. Ege, A. Das, L. Batina, and I. Verbauwhede, “Security of Countermeasures Against State-of-the-Art Differential Scan Attacks,” Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2013), co-located with IEEE European Test Symposium (ETS 2013), May 2013.

Sections 3.1 through 3.8 are based on [A]. Section 3.9 is partly based on the discussion of previous work on scan attack countermeasures in [B] and [C]. Section 3.10 is derived from [C].

Personal Contributions

- One of the principal authors (around 50% contribution in each publication).

Part I: Countermeasures from other side-channel attacks

3.1 Introduction

Cryptographic ICs need to be protected against a wide set of side-channels, such as power analysis and fault attacks. Several countermeasures are reported in the literature [52]. In this work, we analyze the effects of Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Fault Attack countermeasures on the scan-based attacks. We perform a differential scan attack (DSA) on popular SPA and Fault attack counteracting techniques and demonstrate the results. We further analyze the effect of DPA countermeasures in protecting against DSA. In addition, we mathematically model DSA on Public-key cryptographic hardware with different DfT configurations and present results employing commercial digital testing tools. Finally, the side-channel and fault attack vulnerabilities of the countermeasures are summarized.

This part of the chapter is structured as follows: Section 3.2 briefly presents the side-channel attacks and some countermeasures. Section 3.3 presents our novel mathematical modeling of scan attacks and leakage analysis based on various DfT configurations. In Sections 3.4, 3.5 and 3.6, scan attacks on public-key cryptographic implementations with SPA, DPA, and Fault attack countermeasures are described, respectively. A summary of the attack vulnerabilities of different countermeasures is presented in Section 3.7.

In this work, we have investigated the effect of practical scan attacks on hardware implementations of RSA and ECC. The differential scan attack technique described in Chapter 2 are extended with countermeasures against power and fault attacks in place. Mathematical modeling of scan attacks, which was not provided in any previous work, has been proposed in this thesis. Differential scan attacks are performed on a variety of hardware implementations of RSA and ECC with Side-Channel attack (SCA) and Fault attack countermeasures including practical DfT structures. There are numerous countermeasures reported in the literature which are based on different design levels (for example, algorithmic level or circuit-level). Circuit-level countermeasures like gate-level masking and secure logic styles aim to protect the information against power leakages and do not interfere with the scan-based attacks, which exploit the DfT structure. In this chapter, we have implemented some widely used algorithmic countermeasures when performing our proposed differential scan attack and have demonstrated experimental results.

3.2 Side Channel Attacks and countermeasures

A side-channel attack may be defined as the retrieval of secret information out of a cryptographic implementation through the physical side-channels without breaking the underlying primitives [56].

- SCA exploits physical properties of the cryptographic implementation, such as power consumption, time, fault insertion, or Electromagnetic (EM) emission. Timing based side-channel attacks was first presented in [75], while power based side-channels was proposed for the first time in [76]. EM radiation based side-channel attacks were introduced in [53] and [107].
- There are many kinds of SCAs: the two most well-known ones are Simple and Differential.

In Simple-SCA, a single side-channel measurement is taken for a set of inputs, and a statistical analysis is performed to take a decision on the secret key bit. For example, in the case of ECC, this can be achieved by observing the difference in power consumption between the operations of point addition and point doubling, whereas, in Differential-SCA, multiple traces are taken for different input vectors, and a statistical offline analysis is performed to take a binary decision for obtaining the secret key bit. For instance, power traces are collected for thousands of different point multiplications. Then these traces are divided into two groups based on a specific bit of the input base points. The difference of means of these two groups is computed, and a decision is made based on the observed value [67].

The basic principle of countermeasures against Simple-SCA is to implement an algorithm with balanced execution irrespective of the secret bit. Balanced execution requires that the same operation is performed if the key bit is 0 or 1. Some varieties of these countermeasures are Montgomery ladder [70], atomic execution [85], and the unified formula [68] (more details are given in Section 3.4).

Similarly, Differential-SCA countermeasures are mostly based on randomizing the processed data to suppress the correlation between the physical properties and the original data. Some varieties of these countermeasures include blinding the secret exponent [67], blinding the base [67], randomizing the homogeneous projective coordinates [67], random key splitting [27], randomized EC isomorphism [27], and randomized field isomorphism [27]. A detailed survey of side channel attacks and countermeasures appears in [52].

The details of differential scan attacks on PKC has been explained in Section 2.3.

3.3 Mathematical Modeling and Attack Scenarios

3.3.1 Mathematical Analysis of DSA on PKC

For N_p different input pairs, the probability of a bit flip in the intermediate register is:

$$P(N_p) = \frac{2 \cdot N_p - 1}{2 \cdot N_p} \tag{3.1}$$

As explained in Section 2.3.1.5.1, in the presence of advanced DfT structures, not all the bits of the intermediate register are observable. Let N_l be the number of leaking test outputs. The probability that at least one of them satisfies the property required to retrieve a secret bit is:

$$P(N_l) = \frac{(2 \cdot N_p)^{N_l} - 1}{(2 \cdot N_p)^{N_l}} \tag{3.2}$$

Then, the probability of retrieving N_s secret bits with N_p input pairs and N_l leaking test output bits is:

$$P(N_l) = \left(\frac{(2 \cdot N_p)^{N_l} - 1}{(2 \cdot N_p)^{N_l}}\right)^{N_s} \tag{3.3}$$

In the above equations, the length of the scan chains is not considered as a parameter in deriving the attack success probability. Leakage analysis (as explained in Section 2.3.1.5.1) is performed beforehand and makes the attack independent of the length of the scan chains, as the useful leaking slices is found prior to the attack.

Additionally, to reduce the required number of inputs, we can choose a set of inputs $(IN_j, IN_k, IN_x, IN_y, ...)$ instead of a set of pairs (IN_j, IN_k) and generate pairs by choosing all the possible combinations of two elements from the set of inputs. For instance, with 10 inputs we can derive 45 pairs. Therefore, we show in Figure 3.1 the function of required number of inputs against leaking test outputs with 99% of confidence. As can be seen, with 6 leaking test outputs, the attacker needs to know only 3 inputs (where the minimum to generate a pair is 2). This approach is similar to the differential attack procedure employing structures of messages as introduced in [20].

3.3.2 DfT Configuration and Leaking Slices

The number of leaking slices depends on the DfT structure. Higher compaction ratios lead to less leaking slices while no compaction means that all scan flip-flops

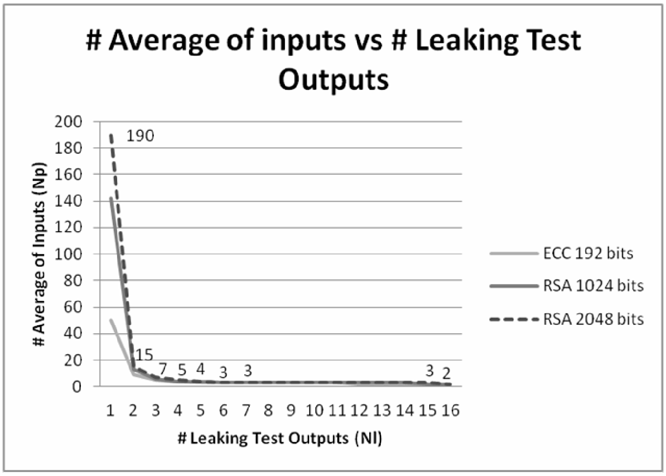


Figure 3.1: Crypto inputs required to perform the attack vs. the number of leaking slices

Table 3.1: DfT Configurations used to analyze the number of leaking slices

DfT Config.	Algorithm	Compaction Ratio	No. of Chains	Length of Scan Chains	No. of Test Outputs
1	ECC	2/1	16	214	8
2	RSA	2/1	32	321	16
3	ECC	8/1	32	107	4
4	RSA	8/1	64	161	8
5	ECC	32/1	32	107	1
6	RSA	32/1	64	161	2
7	ECC	128/1	128	27	1
8	RSA	128/1	128	41	1

are observable at the test output. We implemented two designs corresponding to Algorithms 3 and 4 in Chapter 2, and inserted different DfT structures using the Synopsys DfT Compiler (v2010.12) to count the number of leakage slices. These different DfT configurations are displayed in Table 3.1.

The number of leaking slices also depends on the number of scan flip-flops that store secret information (SFFs). If the attacker knows the details of the design, he can perform the proposed attack on any flip-flop that stores information related to the secret. For instance, the RSA intermediate register A, which has a length of 1024 bits, can be represented as SFFs. Besides this 1024-bit intermediate register, there are also other registers that may be related to the secret. This register can

Table 3.2: Number of leaking slices for different DfT configurations

DfT Configuration	1	2	3	4	5	6	7	8
No. of Leaking Slices	1624	3589	433	608	109	153	28	39

be exploited if the attacker is aware of its presence through the knowledge of the underlying algorithm. The designs considered in this work have 3074 and 4096 SFFs for ECC and RSA, respectively, from a total of 3421 and 10,258 flip-flops.

In order to count the number of leaking slices for each DfT configuration in Table 3.1, we count the slices that contain at least one SFF. Table 3.2 shows the results. The number of leaking slices for high-compaction (Configurations 7 and 8) drops considerably. However, even with compaction ratio of 128:1, the DfT structure has several leaking slices, which is sufficient to perform the attack (see Figure 3.1). Even in system-on-chip (SoC) designs, where multiple IPs need to be tested, the number of leaking slices are not expected to change since it is a common practice to test each IP in isolation to limit the temperature and not burn the circuit.

3.3.3 Handling the Mask Decoder

The DfT configurations displayed in Table 3.1 do not include mask decoders (Section 2.3.1.5.1). The number of leaking slices depends on the mask employed. However, we describe in this section a method to cope with the secret information losses due to X-Masking.

Usually, the mask decoder is controlled by the test inputs, passing through the input decompressor. Therefore, the actual mask applied to the scan chains is controlled by the test engineer and thus by the attacker. Most of the proposed scan-based attacks do not rely on the input patterns: they are based only on the observability of the response vectors. While unloading a response to analyze it, the attacker can also load mask values (valid for DfT structure inserted by Synopsys' DfT Compiler (v2010.12)) and decide which slices to block. In case that no secret information is leaking due to masking, the attacker changes the input patterns until he finds a pattern that produce at least one leakage slice.

In the following sections, we address scan-based attacks on public key cryptographic implementations which have SCA and Fault Attack countermeasures built-in.

3.4 Scan Attacks with SPA Countermeasures in Place

Simple Power Analysis (SPA) analyzes the power consumption curve of a device when a cryptographic operation is being performed. The amount of power consumption varies depending on the instructions being executed. The operations executed for a secret corresponding to 0 are usually different from that executed if the secret is 1. Therefore, by observing the power consumption, the attacker can distinguish if the secret is 0 or 1. In addition, the attacked instruction needs to have a relatively simple or direct relationship with the secret key. For example, SPA can be used to break RSA implementations by revealing the power consumption difference between a multiplication and a squaring, which are directly related to the private key of the user.

In order to protect a circuit against SPA, it is essential to implement a cryptographic algorithm with a key-independent balanced execution. The following approaches are in general used to prevent SPA vulnerability of a cryptographic algorithm.

- Inserting dummy operation: here, some redundant execution is performed for balancing two paths for $k_i = 0$ and $k_i = 1$. Examples of this countermeasure are the Always double-and-add and the Montgomery Ladder [70].
- Atomic execution: this countermeasure sub-divides each path with a number of fixed operation sets. Therefore, the execution always follows a straight line consisting of the same operation sequence. Some examples of this countermeasure are the Atomic square-and-multiply and the Atomic right-to-left algorithm for RSA, and the Atomic double-and-add for ECC [85].
- Unified execution: in this technique, both paths (secret equal to 0 and secret equal to 1) are executed with the same set of operations so that they cannot be distinguished and exploited to guess whether $k_i = 0$ or $k_i = 1$. An example of this countermeasure is the Binary Huff curve-based SPA-resistant ECC implementation [68].

However, an SPA countermeasure does neither change the scan design of the hardware circuit for the respective algorithm nor its intermediate iterative results. Therefore, it is vulnerable to our differential scan attack, as demonstrated in the following subsection.

Vulnerability of SPA Countermeasures against SCA

SPA countermeasures do not protect against scan-based attacks. We first evaluate some up-to-date countermeasures for RSA and ECC. For each implementation of

Table 3.3: Scan attack on SPA resistant RSA implementations

No. of unmasked secret FFs for different countermeasures				Average No. of messages to retrieve the 1024-bit secret exponent
%	ME	ASM	ARB	
100%	512	640	640	2
75%	384	480	480	2
50%	256	320	320	2
25%	128	160	160	2
1	1	1	1	142

these countermeasures, we execute the proposed scan-based attack and verify that it is capable of retrieving all the secret bits. Both RSA and ECC crypto blocks and corresponding countermeasures are emulated in software. We also model a multiple scan chain scenario (with eight chains) with a response compactor that reduces eight bits into a single parity bit. This approach has no impact on the security evaluation and it allows faster simulation times. In order to evaluate the effect of X-Masking, we remove some flip-flops from the scan chains. For that purpose we define five scenarios:

- all intermediate register FFs (IRFs) are present in the scan chain,
- three-fourth (75 %) of the IRFs are inserted in the scan chain,
- half (50 %) of the IRFs are inserted in the scan chain,
- one-quarter (25 %) of the IRFs is inserted in the scan chain, and
- a single FF of the intermediate register is in the scan chain.

Results for the SPA-resistant RSA implementations for different countermeasures are shown in Table 3.3, while that for the SPA-resistant ECC implementations are shown in Table 3.4. The implemented RSA countermeasures are Montgomery Exponentiation (ME), Atomic Square and Multiply (ASM), and Atomic Right-to-left Binary (ARB) algorithms, while for ECC implementations, the countermeasures are Montgomery Ladder (ML), Always Double-and-add (AlD), Atomic Double-and-add (AtD), and Unified Point Addition (UPA).

Since the required number of input pairs is different for each key, we followed the same approach for several keys and obtained an average number of inputs to retrieve the entire secret. As can be seen, the number of required inputs when several FFs are present is less than when only one FF is inserted in the scan chain. It comes from the fact that more unmasked FFs in the scan chain implies more leakage points. Therefore, it is easier to satisfy the good pair of points property shown in Figure 2.16 with more leakage points. Additionally, the number of required points to retrieve the entire secret is the same for different countermeasures (for both RSA and ECC). It happens because the number of secret flip-flops for 100, 75, 50,

Table 3.4: Scan attack on SPA resistant ECC implementations

No. of unmasked secret FFs for different countermeasures					Average No. of points to retrieve
%	ML	AlD	AtD	UPA	the 192-bit secret scalar
100%	360	360	264	312	2
75%	270	270	198	234	2
50%	180	180	132	156	2
25%	90	90	66	78	2
1	1	1	1	1	50

and 25 % is still enough to generate more than 6 leaking slices ($N_l = 6$). Therefore, with only two inputs, all the bits of the secret can be retrieved. In other words, these countermeasures are completely transparent to the differential scan attack.

3.5 Scan Attacks with DPA Countermeasures in Place

DPA is based on the correlation between the processed data and the power consumption. The power consumption of a device varies as the same operation is performed on different data. Therefore, the attacker can observe differences on the power consumption and retrieve the intermediate value stored after the operations, and thus the secret key. But this variation is very small and may not be visible from their direct plots. Thus, in DPA, the attacker records the power consumption of several runs of a cryptographic algorithm with different plain texts, but a fixed secret key. A sophisticated offline analysis is then performed on the recorded power consumption data to reveal the secret key based on statistics.

Various techniques exist to protect a cryptographic implementation against DPA. In general, one class of techniques replaces the original data by random masked data, perform respective cryptographic execution, and at the end it is unmasked to get the actual output. The masking can be applied at algorithmic level as well as gate level. For asymmetric key algorithm all existing solutions are at the algorithm level, which are as follows:

- Blinding the private exponent: here, the private exponent is masked, so that its correlation with power consumption is eliminated [67].
- Blinding the base: when there is an exponentiation operation, the base represents b in b^d for RSA and P for $d.P$ in ECC. In this countermeasure, the base is masked randomly which destroys the correlation between the actual base and the secret exponent [67].
- Randomizing representation of the base: this countermeasure especially acts on ECC, where curve points can be represented in different ways

based on the respective field types and coordinate systems. This countermeasure randomizes the base point with, e.g. (1) Homogeneous projective coordinates [67], (2) Randomized EC isomorphism [27], and (3) Randomized field isomorphism [27].

Vulnerability Analysis of DPA Countermeasures against DSA

A DPA countermeasure blinds (or masks) intermediate results; thus any observation on these values are not correlated with the secret. More precisely, it adds a random value to the original input and performs operations on this candidate instead of the original data. The last operation removes the random part and outputs the expected value. It suppresses the correlation between power consumption and the original data that are intermediate results of a public key algorithm. As described earlier, our scan-based attack is based on the correlation between scan output and the intermediate results. A DPA countermeasure replaces the original intermediate results with some random values which suppresses our expected correlation at the scan output.

If the designer does not insert the DPA mask register into the scan chains, then scan-based attacks are not able to exploit the leakage of secret information through the test interface. However, it implies that the mask generator circuit must have an ad-hoc test solution. In other words, it must be noted that the hardware that generates the random elements required to implement the described DPA countermeasures must not be in the scan path. Otherwise, the random value may be observable and leak through the test outputs. One of the solutions is to use on-chip random number generator (RNG) monitoring units to quickly and efficiently test the quality of the generated random numbers using standard NIST randomness tests [126]. Another solution to test this generator is to use a shadow register to store the actual value used in the blinding. During the functional mode this value is used, whereas in test mode the generator uses a dummy register inserted in the scan chain.

Therefore, the DPA countermeasures are also secure against DSA, provided they are implemented properly as mentioned above.

3.6 Scan Attacks with Fault Attack Countermeasures in Place

Apart from passive attacks like SPA and DPA, the execution of a cryptographic operation may be altered actively and the erroneous behavior of the device can be exploited to retrieve the secret. This is known as fault attacks and is applicable to both symmetric [122] and asymmetric ciphers [22, 18]. In this attack, the challenge is to induce the fault at a specific location and time. A fault can be injected by disturbing one or more of the following functional characterizations of a device: power, operating frequency, temperature, EM radiation, light, Eddy current, etc. There are three major types of fault attacks on ECC (also for RSA) implementations [52]. In this section, we describe existing fault attacks and the respective countermeasures. Further, we show whether such countermeasures are vulnerable against our scan attack.

3.6.1 Safe-Error Analysis

This was introduced by Yen and Joye [70, 135]. Two types of safe-error are reported: C safe-error and M safe-error. In the C safe-error attack, the adversary exploits dummy operations which are usually introduced to achieve SPA resistance through balanced execution of each key-dependent branch. In order to counteract against C safe-error, the Montgomery ladder can be used [70], which provides a balanced execution as well as the results of all operations are accumulated to the final result. However, the experimental results presented in Section 3.4 show that execution of elliptic curve scalar multiplication as well as RSA exponentiation through Montgomery ladder is vulnerable against our scan-based attack. Thus, it is concluded here that the existing C safe-error countermeasure does not guarantee security against scan-based attacks.

On the other hand, in the M safe-error attack, the adversary induces an error during a specific time in the memory elements holding the intermediate results of an iterative execution. Based on this error, the Montgomery ladder shown in Algorithm 4 is vulnerable. Let us assume that it executes point addition followed by point doubling. Further, the adversary induces a temporary fault at Q1 register after starting the execution of point doubling. Now if $k_i = 1$, then this fault will be cleared; else the fault will be there, and it could be observed through the final result. In order to overcome this attack, it is recommended in [70] to execute the point addition and point doubling operations in parallel on two different processors. It prevents the conditional (key dependent) clear out of the induced M safe-error from Q1 register. However, this countermeasure does not affect our scan-based attack scenario. Our attack is based on the final results and the scan out bits both of which are not affected by the above M safe-error countermeasure. Thus, it would be vulnerable with similar success rate as shown in the results of Montgomery ladder in Section 3.4.

3.6.2 Small Subgroup Attack and Curve Integrity Check

An elliptic curve in a large prime field (F_p) is defined by the following equation:

$$y^2 = x^3 + a \cdot x + b \tag{3.4}$$

Solutions to this equation in F_p represent affine points on the elliptic curve. These points form several subgroups, among them, the subgroups with large prime order $\approx$ p which are used in elliptic curve cryptosystems. One major problem of elliptic curve is that it does not use the curve parameter b to compute point addition/doubling and scalar multiplication [52, 51]. Let us consider an ECC implementation of a curve where b_1 is actually used. In order to perform a fault attack, the adversary inputs a faulty point belonging to the curve with b_2 instead of b_1 as a base point in elliptic curve scalar multiplication. Then he exploits the output point with small order to find out the secret scalar.

The potential countermeasure against the above small subgroup attack is known as curve integrity check, which checks the integrity of the input point against the actual curve equation before starting the execution of the elliptic curve scalar multiplication. If the curve integrity check is not satisfied, then the device will terminate the execution immediately and ask for a valid input. Thus, the attacker cannot run the device with a point with small order.

Although this countermeasure protects the elliptic curve secret scalar against the above fault attacks with some additional computation, it does not destroy any features required to perform our differential scan attack on ECC implementations. Thus, our scan attack, as described earlier, has a similar success rate in the presence of this curve integrity check countermeasure.

3.7 Summary of Vulnerabilities and Countermeasures

Table 3.5 summarizes the cross-relationships between the side-channel attacks and the selected countermeasures. The side-channel attacks are divided into three categories: power analysis, fault attacks, and scan-based attacks.

In summary, we can say that randomizing the input data (e.g. the base point in ECC) removes the correlation between he scan-out data and the cryptographic operation being performed and makes the scan data useless for an attacker. Combination of test compression with DPA countermeasures can also help protect against Differential scan attacks. However, care should be taken that the random mask used in DPA countermeasure is not inserted in the scan path.

Table 3.5: Summary of effectiveness of different countermeasures against side-channel attacks for public key algorithms

Countermeasures	Power Analysis		Fault Attacks		Scan Attacks
	SPA	DPA	M-safe error	C-safe error	DSA
Unprotected Montgomery exponentiation	✓	×	×	×	×
Atomic square and multiply [85]	✓	×	×	×	×
Atomic right-to-left binary algorithm [85]	✓	×	×	×	×
SPA-resistant Montgomery Ladder [69]	✓	×	×	✓	×
Always double and add [70]	✓	×	×	×	×
Atomic double and add [85]	✓	×	✓	✓	×
Scalar randomization [27, 67]	×	✓	×	×	✓
Base point blinding [67]	×	✓	×	×	✓
Random projective coordinates [67]	×	✓	×	×	✓
Randomized EC Isomorphisms [27]	×	✓	×	×	✓
Randomized Field Isomorphisms [27]	×	✓	×	×	✓

Legend: ✓: Resistant. ×: Vulnerable

Part II: Countermeasures Specific for Scan Attacks

3.8 Explicit Scan Attack Countermeasures

Some of the major scan attack countermeasures proposed in the literature are presented here.

3.8.1 Insertion of inverters in the scan structure

In the 'Flipped Scan Tree' architecture [120], inverters are introduced at the scan-in input of some of the Scan D Flip-Flops. The location of the flipped scan flip-flops in the scan tree architecture is known only to the designer and the SoC Tester, and completely unknown to an attacker, who cannot interpret the generated test patterns to seek useful information, without this knowledge. This method though not consuming high test area overhead and having low test time, has issues with the communication of the test structure between the designer and SoC Tester. Without this knowledge, testing of the chip with this custom test structure is not possible. This can be interpreted as security by obscurity, which may not be considered a good secure design practice. Moreover, in spite of an unknown test structure, it is still possible to attack this scheme by observing enough number of test inputs and corresponding responses using the differential scan attack strategy, which is independent of the scan chain architecture. The reason for this is that since the position of the inverters is fixed in the scan structure, it is completely transparent to differential scan attacks, where pairs of plaintext inputs are given, and the Hamming distance between the responses are observed.

3.8.2 State Dependent Scan Flip-Flops

The flipped scan tree method has been fixed to a certain extent using the dynamic variable scan approach, through the use of state-dependent scan Flip-Flops (SDSFF), as presented in [95]. The modified scan flip-flops are state-dependent, which would cause the output of each SDSFF to be inverted based on the XOR of the present input to the SFF and the past value of the SFF stored in a latch. The structure of the scan path can change dynamically even after it is designed. There is also a provision for changing the security level flexibly at test time and the scheme does not require a controller for reducing area overhead. This is claimed to make the discovery of the internal scan architecture more difficult. However, it still involves security through obscurity by hiding the location of these special scan flip-flops in the scan structure. Moreover, the security depends on the number of replacements of ordinary scan flip-flops with the state dependent scan flip-flops, and hence involves an area overhead of up to 16% to ensure higher security.

3.8.3 Scan Chain Scrambling

In this approach, the order of the scan chain elements is altered by a scrambler [63]. When the scan mode has been reached securely, the scan chain elements are ordered in a predetermined manner. This is implemented by employing a scrambling controller which generates the control signals of multiplexers inserted between the scan chain elements. However, in insecure mode, the order of the scan chain elements keeps changing at a certain frequency. Each scan chain is divided into multiple scan elements and the order of connections of the scan elements is controlled through the scan chain scrambler. The scan chain scrambling methodology is represented graphically in Figure 3.2.

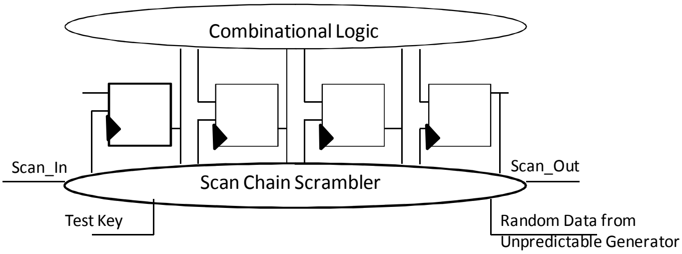


Figure 3.2: Scan Chain Scrambling

The extra area requirement of this scheme is quite small. Though test time can reduce using this method due to reduced size of scan chain segments, routing of the complete design can become more difficult due to modifications in the standard scan path.

3.8.4 Removing All Traces of Secret Information in Test Mode

The approach taken in [61] is to reset the chip and remove all traces of any secret information or cryptographic algorithm execution in test mode. This is achieved through an elaborate reset mechanism which checks in detail for any remnants of secret functional mode data and only allows the circuit to enter the test mode if no such data is present. Here access to test features is not prevented, but it is made useless in retrieving secret information. The chip controller of a standard IEEE 1149.1 Boundary Scan is modified such that once the chip is in the test mode, the activation of the test feature (scan in and out operations) is only possible after an initialization process aimed at protecting secret information. Once the chip is in the normal operation mode, no data from the scan operations remain. This provides a strong security mechanism, but is not suitable for cryptographic applications where the key or any other secret information needs to be stored on-chip.

3.8.5 Modified Partial Scan

This scheme (also known as balanced secure scan) was proposed in [65]. It aims to protect non-scan registers by employing a test controller that enables the test mode only when an authentication succeeds. Only a few flip-flops belonging to the secret registers are included in the scan chains. Further confusion is added to the kernel wherever a secret register is inserted in the scan chain. The partial scan methodology is represented graphically in Figure 3.3.

The outline of the proposed method is as follows. First, scan registers are selected so that the kernel becomes a balanced structure and the number of FFs in secret registers selected as scan registers is minimized. Then, if some secret registers are selected as scan registers, confusion circuits are added into the kernel to randomize the values of the secret registers in test mode while preserving a balanced structure.

As shown in the figure, 'test1' and 'test2' control the functioning of the test controller which in turn decides which mode the circuit will be in. If either 'test1' or 'test2' is high, the circuit is in test mode; if both are low, the circuit is in normal functional mode. In test mode, the shift operation of the scan chain is enabled. Moreover during test mode, a dynamically changing mask is XORed to the key-dependent secret FFs to add confusion to them. Here 'confused signals' actually mean the secret crypto FFs which need to be protected from an attacker. Once the circuit is in normal mode, it cannot be shifted to test mode. This is due to the FFs which maintain their states and their outputs are ANDed with

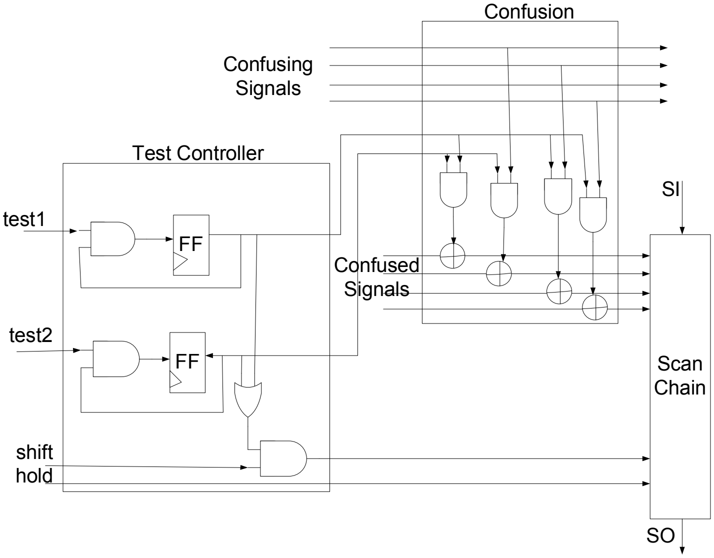


Figure 3.3: Partial Scan [65]

'test1' and 'test2'. There is a finite state machine (FSM) which controls the switch between normal mode and functional mode for different test runs.

Using this method, 100% fault efficiency is demonstrated to be achieved for all the cases considered in [65] (Partial Scan DfT structures implemented on an open-source RSA decryption core, 100%, 50% and 25% confusion added) with reasonable test generation time (for instance, 72.66 sec for the case 50% confusion, instead of 30.47 sec for full scan). However, the method identifies more redundant faults than full scan. The area cost for incorporating this scheme is restricted between 6% and 9% depending on the amount of confusion circuits added.

In spite of the above promising features of the scheme, it can be still targeted by the new differential scan attack approach explained in Section 1.3.7 with close to 100% success rate.

3.8.6 Lock and Key Technique

The Lock and Key technique [77] is intended for preventing malicious attackers from revealing secret information stored in the chip. The scan chains are divided into smaller sub-chains of equal length and a random selection of the sub-chain is made when an unauthorized user attempts to access the scan chains by switching to the insecure test mode. Thus, malicious users cannot predict where in the scan chain the stimuli on the scan inputs go and where the response from the scan outputs come from. Test vectors are not sequentially shifted into each sub-chain but instead a LFSR selects a random sub-chain to be filled. The general structure is represented in Figure 3.4.

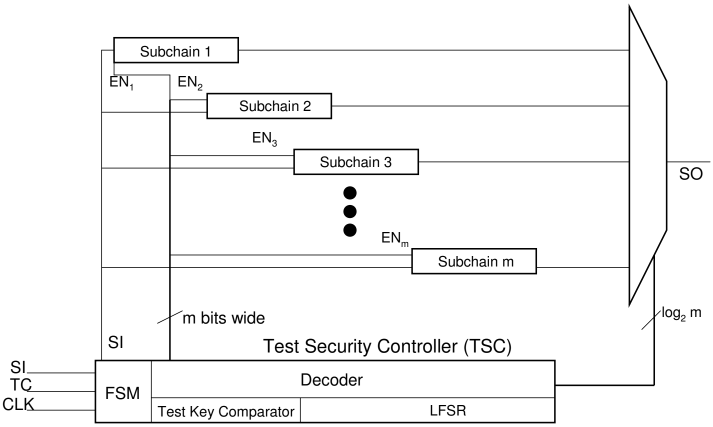


Figure 3.4: Architecture of the Lock and key technique

When the circuit under test (CUT) is initially reset, a Finite State Machine (FSM) sets the Test Security Controller (TSC) into insecure mode and will remain in this insecure state until TC is enabled. It is only after TC has been enabled for the first time and a test key has been entered that the TSC may exit the insecure state. When a test key is entered and a user has been ensured to be a trusted user, the FSM allows the TSC to enter the secure mode allowing predictable operation of the scan chains and will remain in this state until the CUT is reset. Otherwise, the TSC will remain in insecure mode and the behavior of the scan chain will no longer be predictable. If the entered key fails, the TSC remains in insecure mode and will seed the LFSR with an unpredictable random seed, essentially locking the scan chains from being used correctly. Since the choice of sub-chain is pseudo-random due to the LFSR, it is difficult to predict the response on SO if both the seed and the configuration of the LFSR are unknown. In insecure mode, the scan configuration keeps changing with each test clock and is unpredictable. The predictable behavior of the LFSR primitive polynomial is removed when functioning in insecure mode for sufficiently long time. This is done by changing the LFSR configuration in insecure mode by using some additional bits (active only in insecure mode through multiplexers) which changes the feedback to the LFSR. Moreover, the FSM can be configured to generate a new random LFSR seed every round to further make scan attacks difficult. However, LFSRs have a linear structure and generate a predictable sequence of values. Weaker implementations can be broken by crytanalysis using Berlekamp-Massey algorithms [17] or other similar methods.

3.8.7 Design for Secure Test

The Design for Secure Test (DFST) [121] is an ad-hoc solution targeting the round structure of AES chips in particular. Since in AES, the rounds consisting of the operations: Substitute Bytes, Shift Rows, Mix Columns and Add Round Key are identical, they should have identical test responses as well, when fed with the same input test pattern. The responses are then compared in a special comparator for parity. The idea of this scheme, though simple, provides an ad-hoc solution only for AES crypto chips which have a regular and completely unrolled round structure. Such AES implementations have high area requirements. Moreover, this method would not work for testing crypto chips with an asymmetric structure. Hence, this secure test strategy has limited applicability.

3.8.8 Masking Schemes

In [36], two masking countermeasures for protecting AES against scan attacks were proposed. These masking schemes are similar to the countermeasures used to protect against differential power analysis (DPA) based side-channel attacks. The first method is based on masking the round-register data. The mask can be added to all the 128 round-register FFs, and then removed from the encrypted value before executing the next AES round. The masking is effective only during testing and is completely transparent in functional mode. Alternatively, to reduce area requirements, the mask can be applied on a single FF per slice to be protected. In this manner, the parity of the whole slice is affected by the mask and its effects cannot be eliminated by the attacker.

The second method works on modifying the response compactor. This is performed by masking the parity bitstream on the output of the test response compactor instead of the data before being captured in the scan chain. It makes use of an enhanced LFSR (eLFSR) that can function either as a simple register or as an LFSR. In functional mode, the eLFSR is loaded with a value matching with the round-register (a 128 bit length value unknown to the attacker), or with any other value depending on the input message and on a large part of the secret key. In test mode, the eLFSR provides the mask to the observed stream.

These Masking schemes mask each bit of the round register which is the target of scan attacks. Therefore, the scan attacks employed in this thesis do not work against these schemes as all the round register flip flops are masked.

3.8.9 On-Chip Comparison

In [38], a scan protection scheme is proposed which allows testing both at production time and during the circuit's lifetime. The process for this is to first scan in both input vectors and expected responses, and then the expected and actual responses are compared within the circuit. This approach has no adverse effect on test quality and no negative impact on diagnostic of modeled faults. Moreover, it requires a negligible area overhead and does not need authentication of the test mechanism as in some other approaches.

The initial process is similar to normal scan testing. An input vector is fed to the circuit-under-test, and the circuit is run in normal mode for one clock cycle. Then the response is stored in the scan chains. However, contrary to standard scan test, instead of shifting out the response, the tester scans-in the expected responses using an external port which is a modification of the Scan-Out in the standard scan. The obtained test response is compared on-chip and pair-wise with the expected one. When all the captured bits in the scan-chain are compared, an additional comparator output pin is set to 1 only if the whole response vector matched the expected one, otherwise the comparator output is set to 0. The comparison of the observed and expected test responses is performed in a secure bitstream comparator which consists of a few logic gates and flip-flops. Diagnostics of faults is enabled by scanning in test vectors with expected responses corresponding to specific faults. Since this method prevents external observation of responses against chosen inputs, it protects against all kinds of scan attacks.

3.8.10 Self-Test of Cryptographic Hardware

There has been some work on secure self-test of AES cores based on BIST approaches which focus on altering the design of the AES through hardware redundancy. A secure BIST implementation on an AES core based on modifying the structure of control unit and the key generator is presented in [44]. The approach in [66] claims to provide secure test access and side-channel attack resistance to the AES core. However, it has some limitations and overheads. It adds one S-Box for every four in the original AES design using repetitions in 16 S-Boxes. It works only for parallel implementations for AES and has relatively higher area overhead as the S-Boxes consume the largest area of the AES hardware implementation. Moreover, the original AES core is altered and may not be suitable for an SoC integration scenario. There has also been some previous work for secure testing of public-key cryptographic cores using a low-area overhead approach [55]. However, it is only applicable for ECC cores containing the Digit Serial Multiplier as the basic building block. Hence, it may be concluded that any approach working on self-test of the cryptographic core either modifies the existing design or works with specific implementations, and is not suitable for an SoC environment. Moreover, though

BIST solutions maintain security of the circuit under test, they only provide a pass/fail signature which is not useful for diagnosis, and can have relatively higher area overhead.

In this chapter, on the other hand, the proposed secure cryptographic SoC testing approaches are generic and work for all flavors of cryptographic hardware implementations. They do not alter the crypto core or any existing DfT present in the IP module. They are highly suitable for an SoC environment, where the entire cryptographic hardware implementation, with any DfT is provided as a hard IP module for integration on the SoC.

3.9 Differential Scan Attack on Scan Attack Countermeasures

In [37], a new powerful scan attack on AES is presented which is demonstrated to be successful against Partial scan and X-Masking schemes. This is explained in Section 1.3.7. However, in the following sub-sections, we evaluate and compare the applicability of the classical scan attack principle [133, 36, 36] and the new attack [37] when test compression and other advanced DfT features are combined with some of the existing scan attack countermeasures.

3.9.1 Combined Scan attack on AES with Test Compression and Scan Attack Countermeasures in Place

As explained earlier in Section 3.8.1, insertion of Inverters in the scan path is completely transparent to differential scan attacks as the position of the inverters is fixed in the scan chains and their effect is neutralized in differential mode. Some of the other explicit countermeasures such as scan chain scrambling and partial scan are evaluated in the following sub-sections for their scan attack resistance. The analysis is performed on an AES implementation in the presence of advanced DfT features such as test compression, X-tolerance, static and dynamic X-Masking to consider a more realistic industrial scenario.

The scan chain scrambling countermeasure can be emulated by changing the order in which the scan chains are connected using a pseudo-random generator at a frequency which is a fraction of that of the scan frequency. We chose to simulate the effect of this countermeasure by starting from a KFF distribution with 32 active scan chains and 32 active slices, and randomly re-ordering rows and columns of this distribution. In the end, we get a KFF distribution with a different number of active slices and active scan chains. For this work, we generated 1000 random distributions to analyze the effect of this countermeasure.

Table 3.6: Success rates for the attack [49] on test compression schemes with partial scan and scrambling countermeasure

DfT Scheme	Partial Scan - 75%	Partial Scan - 50%	Partial Scan - 25%	Scrambling
X-Tolerant logic	24.1%	5.2%	2.3%	18.5%
Static X-Masking	100%	0%	0%	63.8%
Dynamic X-Masking	100%	0%	0%	53.1%

The results for the attack strategy in Section 2.2.1 are presented in the last column of Table 3.6. Repeating the attack 1000 times took less than 19 seconds in all cases, with the set-up we have used to run simulations.

To emulate the effect of Partial Scan countermeasure in our software implementation of the attack, a random selection of KFFs, according to the chosen parameter, are excluded from the scan design. Results are given for 75%, 50%, 25% masking.

The success rates for the differential scan attack strategy in [49] following the attack principle in [35, 36] is presented in Table 3.6. It shows an average case for an AES implementation containing different test compression logic with a couple of scan attack countermeasures for a random distribution of KFFs included in the circuit.

For X-tolerant logic, as expected, a decrease in success rate is observed with decreasing number of KFFs included in the circuit (for Partial Scan). For Static X-Masking and Dynamic X-Masking, there is a sudden drop of attack success rates from 100 % to 0% when used together with Partial Scan. This can be attributed to the cases where all the subset of KFFs included on the scan chains may be present on a masked scan chain. This result is specific to the DfT structure used in our work. A low non-zero value may be obtained for other possible DfT structures.

However, the new differential scan attack presented in [37] is quite effective against partial scan combined with X-Masking and X-tolerant logic as indicated by the high success rates in Table 3.7. This attack is applied following the methodology proposed in [37]. First, a suitable test input value which leaks information about KFFs is searched. Then, the actual attack is performed by making a key guess, and forming the input set. Later, the input set is processed through the testing circuit in pairs and the resulting test outputs are XORed together. If the output XORs of all pairs from the input set are the same, then the key byte is regarded as the most probable key byte. Whenever the guessed key matches with the actual key of the system, we regard the attack as successful.

Table 3.7 shows that scan chain scrambling method is completely transparent to the attack outlined in [37]. This is because the structure of the scan chain scrambling is assumed to depend on the test inputs. Therefore, as long as the same test input is used for all the elements of the input set, the attack is expected to be successful. Partial scan is almost also broken by the new attack in [37].

The results given in Table 3.7 show the ratio of successful attacks over all 10 000 experiments. There is a dramatic improvement in attack success rates with the new scan attack of [37] illustrating its effectiveness against test compression and X-handling schemes. These results are published in [48]. Repeating the attack 10 000 times took less than 24 seconds in all cases, with the set-up we have used to run simulations.

Table 3.7: Success rates for the attack [37] on test compression schemes with countermeasures

DfT Scheme	Partial Scan - 75%	Partial Scan - 50%	Partial Scan - 25%	Scrambling
X-tolerant logic	100%	100%	100%	100%
Static X-Masking	100%	100%	99.63%	100%
Dynamic X-Masking	100%	100%	99.59%	100%

3.9.2 Scan Attack on RSA in the Presence of Proposed Countermeasures

- In presence of inverters: one of the countermeasures proposed in the literature is the insertion of dummy inverters before some FFs of the scan chain [120]. This technique aims at confusing the hacker, since the sensitive data observed at the scan chain may be inverted. However, since these inverters are placed always at the same location in the scan chain, they are completely transparent to the differential mode.

  The effectiveness of the attack against this countermeasure was validated on the RSA design containing multiple scan chains and compaction/compression module. Two implementations were considered with 4630 and 6180 inverters (50% and 75% of the overall 9260 FFs in the design respectively) randomly inserted in the scan chains. For both cases, the scan attack tool outlined in Section 2.3.1.6 was able to find leakage points and then to retrieve the secret key.

- In presence of partial scan: depending on the design, not all the flip-flops need to be inserted in the scan chain in order to achieve high testability. As proposed in [65], partial scan may be used for increasing the security of a RSA design against scan attacks. However, the authors suppose that the attacker needs the whole sensitive register to retrieve the secret key. As was described earlier, the leakage analysis feature can be used to find out which bits of the sensitive register are inserted in the scan chain. Once these bits are identified, the attack can proceed with only partial information, since each bit of the sensitive register is related to the key.

For evaluating the strength of partial scan, we configured the DfT tool in such a way so as to not to insert some of the sensitive registers (25%, 50% and 75%) in the scan-chain. The tool was able to correctly identify all the leaking bits and then to retrieve the secret key. Also in the worst case situation, i.e., where only one secret bit was inserted in the chain, the tool was still able to find out the correct secret key.

3.9.3 Scan Attack on ECC in the Presence of Proposed Countermeasures

- In presence of inverters: The effectiveness of the attack against this countermeasure was validated on the ECC design with Config.3 of Table 2.6 in Chapter 2. Two implementations were considered with 1677 and 2516 inverters (50% and 75 % of the overall FFs in the design respectively) randomly inserted in the scan chains. For both cases, the tool was able to find leakage points and then to retrieve the secret scalar.
- In presence of partial scan: For evaluating the effectiveness of differential scan attacks on ECC in the presence of partial scan, the DfT tool was configured in such a manner that some of the SFFs were not included in the scan chains. In the first case, half of the SFFs were inserted in the chain. The tool was able to correctly identify all the leaking bits and then to retrieve the secret scalar. Also, similar to the case of RSA scan attack, when only one secret bit was inserted in the scan chains of the ECC design (worst case scenario) with test compression, the tool was still able to deduce the correct secret scalar.

3.10 Noise Injector Countermeasure

The existing scan attack countermeasures (Section 3.8) aim at securing uncompacted scan chain structures. However, as described in the Chapters 1 and 2, industrial DfT structures consist of test compression techniques which lead to a loss of observability of internal scan chains. Hence, they may be extended into a countermeasure. In this regard, we propose a new scan attack countermeasure at gate level based on randomization of the response compactor outputs.

3.10.1 Design of the Noise Injector

Fig. 3.5 shows the proposed noise injector countermeasure. It consists of a Linear Feedback Shift Register (LFSR), a True Random Number Generator (TRNG) and some basic logic gates. For the particular case of the Noise Injector depicted in the figure, two OR gates, one NOR gate, one AND gate, and one XOR gate are required. The LFSR makes a selection of the test cycles when truly random noise is injected into the compactor outputs. More specifically, the compactor output is flipped if 'A' in Fig. 3.5 is 1. The signal 'A' is generated through a combination of TRNG and the LFSR outputs, making it unpredictable for an attacker. Hence, the scan out becomes random and cannot be exploited by differential scan attacks.

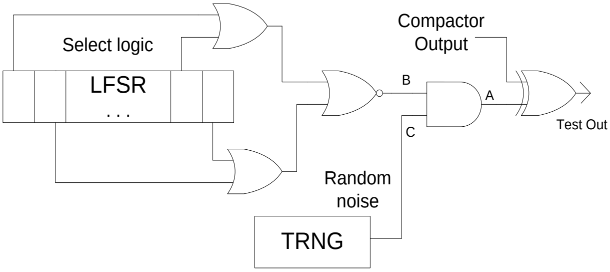


Figure 3.5: Noise Injector countermeasure for Injection Freq. Factor of 16

Injection frequency factor (IFF) refers to the rate at which noise is injected into the compactor outputs, calculated as a factor by which the test clock frequency is divided. As an example, let us consider the LFSR structure in Fig. 3.5 having a IFF of 16. Let us assume that the state bits of the LFSR are completely random, with equal probability of occurrence of ones and zeros ($\frac{1}{2}$). An OR gate will give an output of 0 with probability of $\frac{1}{4}$. Hence the probability that both inputs of the NOR gate are 0 (to give an output of 1) is $\frac{1}{16}$ (as the events are independent of each other). Only when the output B of the NOR gate is 1, true random noise is injected into the compactor output, as the output of the TRNG and B are connected through the AND gate. Similarly to obtain an injection frequency factor of 4, one of the OR gates may be removed and the other input of the NOR gate connected to ground (0 logic), resulting in a probability of $\frac{1}{4}$ for obtaining a 0 at both inputs of the NOR gate, which in turn would cause the noise injection to occur. Similar simple structures can be designed for all the other possible cases. The gate combination to obtain an injection frequency factor which is not a power of 2 involves a larger number of gates. For instance, to obtain an injection factor of 5, we can construct a 4-input truth table which gives five 1s and eleven 0s at its output. One such possible expression would be $(\neg I_1 \wedge \neg I_2 \wedge I_3) \vee (I_1 \wedge I_2 \wedge \neg I_3) \vee (I_0 \wedge \neg I_1 \wedge I_2 \wedge I_3)$, where I_0, I_1, I_2 and I_3 represent the states at the four tap points of the LFSR.

3.10.2 Security Analysis of the Noise Injector

We have also analyzed two ways of attacking this system to determine the security of the proposed method. The first attack does not use any information about the countermeasure, therefore taking a black-box approach. The attack is applied just as it would be applied to any other countermeasure. Results of this scan attack success rates for the noise injector is provided in the second column (Success Rate^1) of Table 3.8. However, there is another way of attacking this countermeasure. An attacker can first give the same input to the crypto algorithm and the same test input to the test circuit. After collecting the outputs by repeating this procedure a sufficient number of times, the attacker can figure out the points where the TRNG corrupts the output. Later, DSA can be applied to the circuit with the same test input and these bits can be removed from the test output as they have the potential to corrupt the test output. We simulated the attack in software and the probability of such an attack to be successful is presented in the third column (Success Rate^2) of Table 3.8. A number of different settings are experimented with and they are listed with respect to the frequency of true random bit injection to the compactor output.

As is evident from the table, the success rate of the attack decreases drastically when noise is injected at a higher frequency. Attack success reduces to only 0.9% when noise injected at the same frequency as the test clock.

3.10.3 Impact on Test Coverage and Overheads

Test coverage of the circuit is not affected by the proposed countermeasure. This can be achieved through the following procedure:

- The LFSR structure (feedback polynomial, output points and seed value) should be known to the tester.
- The test cycles should be ignored when signal ‘B’ is 1 (Fig. 3.5).
- For achieving complete test coverage, the test patterns for ignored test cycles should be repeated until signal ‘B’ becomes 0 (sustained vector technique).

The knowledge of the TRNG outputs is not required by the tester, as the test cycles when signal B = 1 are always ignored irrespective of signal ‘C’. We suggest to have dedicated test outputs while incorporating our proposed countermeasure for ICs with redundant pins. In case of pin-constrained applications, the test outputs may be multiplexed with some of the primary outputs. To make the proposed scheme compatible for such applications, an additional control circuit is required which configures the whole cryptographic circuit based on its mode of operations. In

Table 3.8: Change in success rate VS how frequently a random bit is XORed to the compactor output

Injection Freq. Factor	Success Rate[1]	Success Rate[2]
16	63.33%	81.37%
15	59.17%	78.70%
14	54.37%	75.53%
13	49.34%	72.79%
12	43.78%	68.93%
11	37.71%	66.31%
10	32.76%	60.57%
9	28.52%	56.99%
8	23.55%	52.53%
7	18.77%	47.70%
6	15.05%	40.62%
5	11.91%	33.02%
4	8.32%	23.04%
3	5.51%	12.85%
2	2.93%	3.38%
1	0.90%	0.00%

normal mode, the noise injector is not connected to the compactor outputs, while in test mode it is connected. This could be realized by multiplexers controlled by the mode selection input pin.

A TRNG is used instead of a LFSR for generating the random input C as an LFSR has a linear structure which is prone to cryptanalysis if the seed and feedback polynomial is known or if the LFSR does not have sufficient length (incurring high area overhead). A TRNG has much higher unpredictability property. The TRNG used in this scheme is based on Fibonacci and Galois Ring Oscillators [57].

Compared to some of the countermeasures mentioned in the previous section, our proposed scheme has lower area overhead, as we are utilizing the existing test compression infrastructure. Our noise injector countermeasure requires an area of 106.75 Gate Equivalents (GEs) incurring an overhead of 0.61% over the table-lookup based S-box AES implementation in [3] (which needs 17484.25 GEs) with implemented DfT, using a Faraday 130nm library and synthesized using Synopsys Design Compiler version C-2009.06-SP3. The TRNG may also be part of a Cryptographic SoC as an on-chip random number generator. In such a case, it would not require any additional hardware resources. Test application time will however increase by 1/IFF, where IFF represents the Noise Injection Frequency in Table 3.8.

3.11 Conclusion

This chapter introduces an analysis of the effects of SPA, DPA, and fault-attack countermeasures on scan-based side-channel attacks. We implement up-to-date SPA and Fault attack countermeasures for ECC and RSA and verify that differential scan-based attacks (or DSA) are able to retrieve the entire secret. We perform these analysis on gate-level netlists with advanced DfT structures and conclude that those countermeasures are transparent to the proposed DSA. On the other hand, DPA countermeasures usually mask the intermediate values stored in the cryptographic device's registers and therefore also counteract DSA.

Moreover, we evaluate various explicit scan attack countermeasures proposed in the literature for their security properties and implementation overhead. Scan attack results on combined test compression and attack countermeasures are presented. Finally at the gate level, a novel noise injector countermeasure is proposed and analyzed. The analysis of the countermeasures is done in terms of security, testability, area and time overhead in order to determine the trade-off between these secure testing objectives.

Chapter 4

Secure Test Infrastructure

Publication Data

The material in this chapter is based on the following publications:

[A] A. Das, J. Da Rolt, S. Ghosh, S. Seys, S. Dupuis, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Secure JTAG Implementation using Schnorr Protocol," Journal of Electronic Testing: Theory & Applications (JETTA), Springer Science and Business Media New York 2013, Inc., vol. 29, no. 2, pp. 193-209, 2013.

[B] A. Das, M. Knezevic, S. Seys, and I. Verbauwhede, "Challenge-Response based Secure Test Wrapper for testing Cryptographic Circuits," IEEE European Test Symposium (ETS 2011), May 2011.

[C] A. Das, U. Kocabas, A-R. Sadeghi, and I. Verbauwhede, "PUF-based Secure Test Wrapper Design for Cryptographic SoC Testing," IEEE/ACM Design Automation and Test in Europe (DATE 2012), pp. 866-869, March 2012.

[D] D. Karaklajic, A. Das, and I. Verbauwhede, "Secure Mutual Testing Strategy for Cryptographic SoCs," 9 pages, COSIC Internal Report.

Section 4.2 is derived on [A]. Sections 4.3.5 and 4.3.6 are based on [B] and [C] respectively. Section 4.4 is derived from [D].

Personal Contributions

- Principal author.

4.1 Introduction

As presented in Section 3.8, various countermeasures have been presented in the literature to protect against scan attacks. They can be classified into the following categories: structures that are offered by DfT tools and may be considered as inherent countermeasures; protocol countermeasures which change the test procedure in order to secure the test access; and countermeasures that resist against micro-probing of the scan signals. Protocol countermeasures can again be sub-divided in two groups: countermeasures that are extensions of standard test access ports such as JTAG and other countermeasures that secure the circuit at protocol-level.

The proposed scan attack countermeasures in this chapter are grouped into different design abstraction levels: protocol-level, system-level and SoC-level countermeasures. The secure JTAG implementation presented in this chapter aims to protect a cryptographic circuit both at the protocol and system levels by allowing only authenticated users to access the internal data registers through the JTAG port of the system under test. Secure Test Wrapper (STW) approaches are also proposed in this chapter, which guard against scan attacks at the protocol and SoC levels. They are based on a challenge-response based entity authentication protocol. Two variants of this approach are described: a lightweight block cipher based STW and a physically unclonable function (PUF) based STW. Another scan attack countermeasure proposed at the SoC level combines secure test wrappers with efficient multiplier-based BIST. Thus, this test approach intertwines the high security of BIST with the highest testability of scan chains at low area and timing overheads. Together with the proposed gate-level and algorithmic countermeasures presented in Chapter 3, these proposed countermeasures can help protect a cryptographic implementation against security breaches by providing several layers of security defense mechanisms.

4.2 Secure JTAG

We present here a system Level countermeasure in the form of a secure JTAG architecture.

4.2.1 Introduction and Motivation

Joint Test Action Group (JTAG) is the common name for what was later standardized as the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture [6]. JTAG has remained as the ubiquitous test and debug interface standard for circuits and printed circuit boards in the semiconductor industry for more than two decades. The companion standard, IEEE Standard 1532 (Boundary-Scan-Based In-System Configuration of Programmable Devices) has extended JTAG to support on-board programming [12]. A current IEEE standard proposal (P1687, also known as Internal JTAG) seeks to further enhance JTAG by allowing block transfer of data and special instruction sets in order to speed up In-System Programmability.

JTAG is mainly used for manufacturing and in-the-field test-and-diagnosis of VLSI circuits and boards. It may be disabled in the chip-die after initialization of a product. However, there are some applications where JTAG is kept enabled for code or firmware updates. Especially, in case of reconfigurable devices like set-top boxes, where even remote reprogramming may be done through JTAG port based on updates received from the service provider. For instance, the STi7101 low-cost HDTV set-top box decoder and the TI MSP430 used in some set-top boxes have the JTAG open for product support and service.

JTAG was initially designed without a concern for security. As the capability of hardware attackers is increasing, more and more side-channels are discovered, which can compromise the security of a device. One such important side-channel is the improper use of the JTAG port. There have been many practical attacks on secure devices such as set-top box (STB) decoders using the JTAG interface [5]. ARM11 (Cortex) microcontroller, which is used in latest smartphones, has extensive test and debug facilities through the JTAG port. This is a well-known backdoor that is currently used for instance to jailbreak iPhones/iPad, or to unlock protected services in mobile phones [58]. Even if not documented, it is reasonable to think that JTAG could be used to compromise the security of other applications such as mobile e-payments, or Wireless Sensor Nodes (WSNs) [16, 60].

Another security flaw due to JTAG is related to FPGAs. The configuration bitstream which contains the Intellectual Property (IP) information of a reconfigurable design is mostly programmed via the JTAG interface into FPGAs [9]. The firmware update of set-top boxes used in pay-TV subscriptions also happens in most cases through the JTAG port. An insecure JTAG access would allow on one side to re-program parts of the system at the hacker's will, and on the other side it could be used to sniff configuration bits thus allowing retrieving the IP information.

Though there are several approaches for securing the JTAG interface, which can be found in the literature [103, 115, 28, 105, 104], most of them are based on symmetric-key approaches. They have an inherent key management problem. This is what we intend to overcome through the use of Public-key Cryptography (PKC) in our secure JTAG scheme. Though there is previous work on a protected JTAG scheme using ECC-based authentication protocol [24], the scheme uses PKC in a non-standard way causing key-management problems. Moreover, the paper also does not present any timing or area results. Though several PKC protocols can be used for establishing a secure authentication for JTAG, we use the ECC-based Schnorr protocol which is an efficient and secure protocol [118].

In our work, we seek to provide security features to the IEEE 1149.1 JTAG interface by including a Schnorr-based secure test protocol, and present an efficient hardware implementation of the protocol using elliptic curve cryptography. Moreover, our approach does not make any modifications to the existing JTAG interface. To the best of our knowledge, this is the first work that proposes a mechanism for mutual authentication between the secure device and the tester based on a well known and studied public key authentication protocol. Earlier work is either based on symmetric-key systems or only proposes one way authentication, limiting the scenarios in which these systems can be used. The area requirement to incorporate this secure test infrastructure on the JTAG has been optimized to increase the scope of our proposed scheme in a wide range of application scenarios.

Moreover, in this work, we solve the inherent key-management problem of existing Symmetric-Key Cryptography (SKC) based secure JTAG approaches using Public-Key Cryptography (PKC). Specifically, if SKC is used for securing JTAG, there will be a common master secret key for all products or a large secret-key database needs to be maintained at the tester/updater side, which are not good options for mass electronic products. PKC implementations are inherently more hardware expensive and slower than SKC based approaches. Therefore it is a challenging task to incorporate PKC in a resource constrained environment like JTAG.

The use of asymmetric primitives and the related public/private key pairs substantially reduce the complexity involved in key management in this setting of tester against the device. If we take for example the automobile industry, then we expect to bring our car to virtually any garage in the world and get our car serviced. Servicing cars now also includes updating software in one of the on-board units (OBUs) which may be through the JTAG interface. Currently these updates can be pushed to the OBU as soon as it is powered on; no other security measures are used. One of most important reasons for the current lack of authentication is the fact that it presents car manufacturers with a large key management problem that is inherent to the use of symmetric solutions in large scale systems. In symmetric solutions, the verifier needs a copy of the same key that was also used to generate the authentication token (e.g., a message authentication code or MAC on the firmware). This implies that the use of a single master key is very risky as it will be wide spread in many devices and likely to leak at one point in time. Therefore, symmetric-key based solutions require unique keys to be installed at every verifier. In large scale systems, this would require a large database that link the identity of the prover to its key and a means for verifiers to securely access and authenticate this service. Alternatively, key derivation schemes could be used, but they only lower the risk related to a single master key. There are many other application scenarios where similar key-management problems can occur.

To overcome this problem, the solution proposed in this work offers the possibility of using certificates instead of shared symmetric keys. This would for example allow the use of the same signed firmware update for a wide range of OBUs, without the risk of installing the same symmetric key in a range of devices. They just need a valid copy of the manufacturer's public key and a valid digital certificate for signature verification.

4.2.2 Existing Secure JTAG Approaches

An ordinary JTAG standard [6] consists of a pre-defined interface, containing a serial input called TDI, a serial output called TDO, an input for the clock TCK and a mode select input called TMS. By controlling the TMS signal, the user can travel between the 16 states of the JTAG finite state machine, shown in Figure 4.1. Then, the request for executing the instructions and the transfer of data between the circuit and the host is performed by connecting the input TDI and the output TDO to internal shift registers. Thus a malicious host can manipulate the JTAG inputs and execute any instruction.

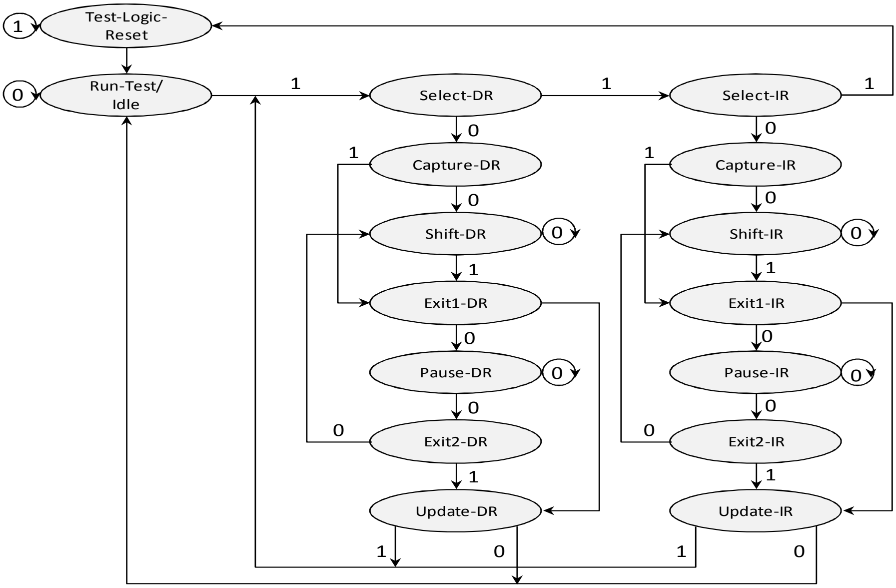


Figure 4.1: 16-cycle JTAG TAP Controller State Diagram

One of the first approaches for implementing a secure JTAG appears in [103]. It presents a locking/unlocking mechanism for controlling the access to the JTAG instructions. It is based on storing a secret key inside the chip boundaries. To gain access to the JTAG features the user must shift in the secret key, otherwise the JTAG bypasses all the data on the TDI input to the TDO output. The scope of this approach does not consider the case where a fake circuit requests code or firmware updates which may compromise the intellectual property.

A detailed evaluation of the JTAG test standard, its security problems, attackers' capabilities, possible attacks and countermeasures has been done in [115]. It presents a JTAG security protocol using a stream cipher (Trivium), hash function and a message authentication code. The authors assume that the service server is trusted, performing one-way authentication. However, to protect the data from unauthorized servers, the data is encrypted.

An anti-tamper JTAG Test Access Port (TAP) is described in [28] that uses SHA-256 secure hash and a true random number generator (TRNG) to create a low gate overhead challenge/response based access system employing an on-chip internal JTAG P1687 instrument. It is mentioned by the authors that malicious designers could modify the designs in order to observe the secret key, implying a one-way authentication scenario.

A multi-level security access system for controlling access to individual scan cells for preventing malicious opcodes from being loaded into the JTAG controller is presented in [105]. This approach also supposes the design is trusted, and thus it is not possible for fake circuits to obtain proprietary updates.

An elaborate three-party secure JTAG protocol using certificates involving SHA-1 hash algorithm, AES block cipher and several arithmetic operators is presented in [104]. The authors describe the possible attack cases, but the protocol is not proven to be secure.

There are also industrial solutions for providing security to the JTAG interface. The Secure JTAG Controller (SJC) which features in Freescale Semiconductors i.MX31 and i.MX31L Multimedia Applications Processors is one such example. It allows for four different JTAG security modes from no security to maximum security. The access to JTAG test and debug features is restricted by a secret-key based authentication mechanism. The security modes are configured using electrical fuses, with the fuse burning being an irreversible process. Similarly there are tools available from various vendors such as Discretix and Lauterbach. TRACE32 PowerTools from Lauterbach provide a Secure JTAG Debug module giving OEMs a secure and authenticated way to debug SoC errors throughout a system's lifetime. It employs the ARM TrustZone technology to provide security at the software level. A detailed overview of the JTAG related fuses and security features in the AVR microcontroller can be found in [7]. Some use-cases and application scenarios involving JTAG security are presented in [113].

To the best of our knowledge, the work in [24] is the only JTAG security solution that is also based on asymmetric key cryptography. In contrast to our work, this solution only provides one-way authentication from the test server to the JTAG device. Moreover, this approach does not improve key management related to symmetric solutions, as it requires the test server to have secure access to a database that contains all the unique private keys related to each device. Because of the non-standard setup of the authentication protocol, every JTAG device must have a unique private key that is stored in a database. This key has to be retrieved by the test server in order to authenticate itself to the device. In our solution, we employ a standard use of public/private keys in which the prover uses its own private key and a certificate signed by a CA to prove its authenticity and not a private key related to the verifier as in [24].

Most of the previous approaches [24, 28, 103, 105, 115] suppose a one-way authentication, where either the circuit or the server is considered trusted. In this thesis, we propose a suitable solution in cases where neither the circuit nor the server is trusted. Additionally, most of the communication protocols in the previous solutions are not as secure as the Schnorr protocol used in this thesis. The Schnorr protocol is proven to be secure under a passive (eavesdropper) attack based on the discrete logarithm assumption [118].

4.2.3 Attacker Model

We have considered the following application scenarios for our attacker model. The JTAG interface of a VLSI circuit is normally used for testing the device, as well as for updating the internal code and firmware in some applications. We assume that the external JTAG interface of the target device is enabled and is accessible to the attacker. In [115], the attacker models are described based on malicious IP cores inside a SoC. However, in this work, we have considered the following two attacker scenarios where internal IP cores are assumed to be trusted, and the attacker is an external entity to the cryptographic SoC. We assume that it is impossible to extract the stored private key (present in secure memory) on the device containing the JTAG interface.

Manufacturing Test/Firmware or Code Update at manufacturer's end: We have considered the manufacturing environment to be controlled and the manufacturer's test server to be trusted. The device may be a fake one (or a clone) trying to get unauthorized code or firmware updates through the JTAG interface. The test server should allow only genuine devices to have access to the updates. Hence, in this scenario, a one-way entity authentication of the device to the Test Server is required.

The device needs to prove to the Test Server that it is in possession of the correct private key, without revealing it to the server. This is achieved through the use of the Schnorr protocol to be employed in our secure test scheme. Here the prover

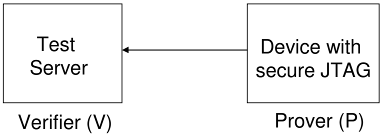


Figure 4.2: Manufacturing Scenario

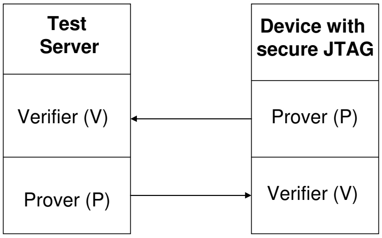


Figure 4.3: In-the-field Scenario

is the device with secure JTAG, while the verifier is the Test Server. This is represented symbolically by the block diagram in Figure 4.2.

A possible use-case for this scenario is system integration, where the integrator procures VLSI chips from different third-party vendors. He needs to make sure that each chip is a genuine one, and not a fake one or clone which can compromise the integrity of the complete system. This can be achieved through the addition of a security feature to the JTAG using an authentication mechanism.

In-the-field Update, Debug and Test: When devices are deployed in-the-field, the environment is considered uncontrolled and both the Test Server and the device with the JTAG may be potential attackers. Hence, mutual entity authentication is required between the device and the Test Server. The Test Server might be malicious and trying to extract the internal secrets from the device through the test infrastructure. Similarly, the device may be malicious or even a fake one, trying to procure unauthorized code or firmware updates through the JTAG interface.

Hence, both the device and the Test Server need to prove their identity to each other without revealing their secrets (their private keys). For the mutual authentication using ECC based Schnorr protocol, when the Secure JTAG is the prover, the Test Server is the verifier. Similarly when the Test Server is the prover, the Secure JTAG is the verifier. This is represented graphically by the block diagram in Figure 4.3.

A possible application of this attacker model is the firmware update of set-top boxes used in pay-TV subscriptions. The user of the set-top box might be a potential attacker trying to get an unauthorized update from the server using the JTAG port to watch pay channels for free. Similarly, an unauthorized update from a remote hacker using the JTAG port might compromise the secret keys stored in the smart card of the set-top box.

4.2.4 Schnorr Protocol

The Schnorr protocol [118] is a zero-knowledge protocol designed for convincing the verifier of the validity of a given statement, without releasing any knowledge beyond the validity of the statement. Using the protocol, a sender can prove his identity to a verifier by showing him that he knows a secret key without revealing it. A system using Schnorr protocol can only be attacked by extracting the private key through side-channel attacks, leaks during installation/generation of the keys, attacks on the CA facilities, etc. In this chapter, we present an efficient implementation of the Schnorr protocol that makes its use cost effective for low cost JTAG devices. Side channel attacks on this implementation or attacks related to software bugs, etc. or not in scope of this work. We claim that our solution is secure in the two scenarios described above, as it is a straightforward use of the Schnorr protocol that is proven secure.

4.2.5 Proposed ECC Based Schnorr Authentication Protocol

We use an enhanced version of ECC-based Schnorr Protocol [118] as the public-key cryptographic protocol in our secure JTAG test scheme. Various public-key implementations, such as RSA or ECC, may be used to solve the key management problems present in previous secure JTAG approaches. We chose ECC as it offers the same security as RSA, with much smaller area footprint. Area overhead is of critical importance, since we are constrained in terms of silicon area required to incorporate security features into JTAG, owing to the small test interface available in most applications. Similarly, various protocols using ECC may be used.

4.2.6 Public Key Verification

When using public key cryptography for authentication purposes, it is essential to verify the authenticity of the prover's public key. Traditionally, the link between a user's public key and some identifier of the user is captured in a digital certificate that is signed by a trusted third party (e.g., certificate authority or CA). By verifying this certificate, a verifier is assured that the public key that is provided by the prover is genuine. This means that it is sufficient to have a copy of the CA's public key in order to verify all public keys that are certified by the CA. It is clear that storing a single CA's public key is far more practical than storing a collection of symmetric keys that are shared with each possible prover. Therefore, we argue that our protocol, although more resource consuming, does provide a more practical solution when compared with previous JTAG authentication mechanisms that are based on symmetric cryptography only.

We propose two modes of this public-key verification, one is purely offline and the other uses an online connection to a trusted Authentication Server (AS). Every verifier has a copy of the CA's public key. Before the actual Schnorr authentication protocol, the prover sends his certificate to the verifier. The verifier simply uses the CA's public key to verify the certificate. In case the verifier has access to a clock, he can also check the expiration date inside the certificate. In case the JTAG device is the verifier, this clock will probably not be available and no expiration date can be verified. Note that in this scenario, it is not possible to revoke certificates, as it is not possible to use an online server to obtain revocation lists or use an Online Certificate Status Protocol (OCSP) like protocol.

In the offline mode, we assume that every prover has a certified public key and this certificate is signed by a trusted CA. In case the verifier has the possibility to contact the online trusted AS, we propose to use a simplified version of the OCSP protocol. The detailed steps of the protocol can be found in Appendix A.

The signature scheme required for the public-key authentication protocol can be implemented using Elliptic Curve Digital Signature Algorithm (ECDSA) [29]. This consumes less area overhead than a 1024-bit RSA signature scheme. An area-efficient implementation is presented in [73]. In our implementation, we have modified the ECC Schnorr controller to allow ECDSA. The hashing involved in the ECDSA signature verification is avoided as we use 192-bit signatures (the same length as the message that is signed, which is the public-key of the prover). Through this public key certificate we protect the Schnorr protocol from man-in-the-middle attack too. Devices which have adequate resources (online connection to the authentication server) to support this authentication process can opt for an online mode, while other devices can have an offline mode of authentication.

In this work, we provide two different implementations. One is over projective coordinates and another is over affine coordinates. In the first design, we do not implement an inversion module whereas it is included in the second design. Due to projective coordinates the first design invokes very few inversions which are performed iteratively on a multiplier unit following Fermat's little theorem. However, in affine coordinates inversion is performed at every iteration of the point multiplication algorithm. Thus, a dedicated inversion unit based on extended Euclidean algorithm is implemented. Here we provide implementation details for both designs which provide better design variations and the user can opt for one of them in practice.

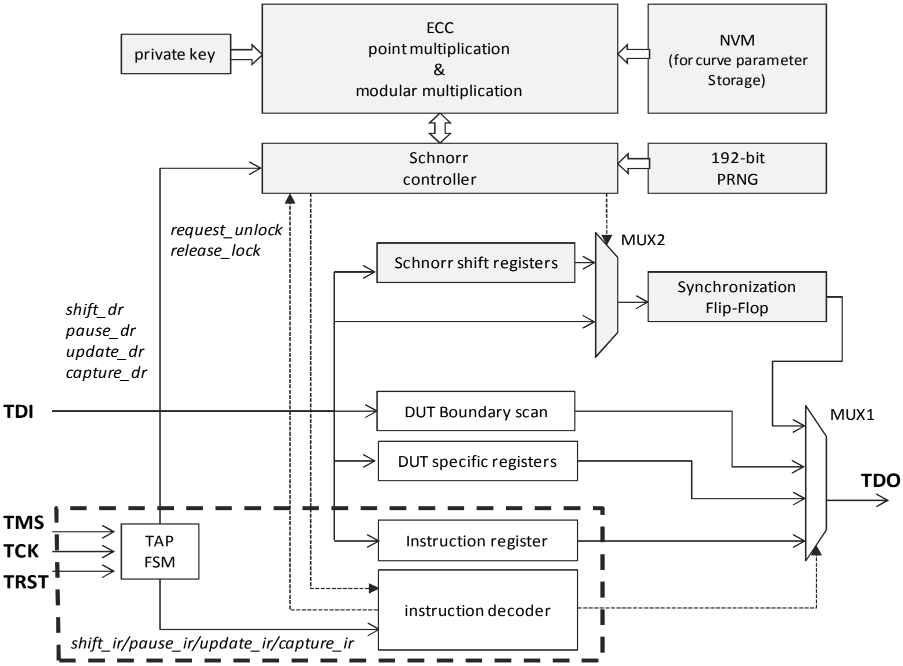


Figure 4.4: JTAG-ECC controller Integration Architectural Block Diagram

4.2.7 Hardware Integration of JTAG with Schnorr Controller

An important contribution in our work is the integration of the ECC based Schnorr controller and ECC point multiplier with the JTAG interface along with the other modules. This has been done in a seamless manner so as not to affect the timing aspects of the IEEE 1149.1 JTAG standard, and also keeping the behavior of the JTAG TAP finite state machine [4](illustrated in Figure 4.1) unchanged.

Our proposed integrated Secure JTAG architecture is shown in Figure 4.4. The ordinary JTAG circuitry is enclosed within dotted lines, and it is divided into its two main components: the TAP finite state machine and the instruction decoder. The Schnorr protocol (described in Appendix B), as well as the ECDSA signature authentication are performed by the Schnorr controller and are placed in the center of Figure 4.4. It interacts with a modified JTAG instruction decoder, ECC module, and a 192-bit pseudo-random number generator (a Linear Feedback Shift Register). The base point coordinates (curve parameters) are fetched from an external non-volatile memory.

The system is locked at the beginning. This is achieved using the Schnorr controller which prevents the instruction decoder from issuing instructions that allow shifting of the internal data registers. In order to unlock it, the tester must manipulate the JTAG inputs to enter the new 'UNLOCK' instruction. Then, the instruction decoder informs the Schnorr controller to start the protocol, by means of the 'request_lock' signal. As soon as the authenticity of the test server is verified, the Schnorr controller activates the 'release_lock' signal, informing the instruction decoder that other instructions can now be performed. For instance, if the system is unlocked, the design under test (DUT) boundary scan register can be controlled. Meanwhile, when 'release_unlock' signal is not active, the instruction decoder sets the multiplexer 'MUX1' to always select the output from the multiplexer 'MUX2', which is controlled by the Schnorr controller, impeding the shift out of any DUT specific register.

During the protocol execution, the communication with the test server consists of using the Schnorr shift registers (192 bits) to shift in and out information required for the protocol. For instance, the transmission of the intermediate values, 'T_a' and 'T_b' (Protocol in Appendix B) is performed by means of shifting out the values 'T_a' (or 'T_b') once the ECC point multiplication is finished. It is important to notice that the shifting is always controlled by the test server, and that the timing for executing point multiplications depends on the scalar multiplier. It means that the Schnorr controller must inform the test server that it has finished each operation of the protocol. This synchronization is achieved by always adding one flip-flop at the end of the Schnorr shift register that is set to '1' if the information in the shift register is valid, otherwise the multiplexer 'MUX2' selects the TDI input and the synchronization flip-flop is set to '0'. Thus, the test server keeps on shifting at least this one bit to detect that the Schnorr controller is ready for receiving the next data. [40].

4.2.8 Implementation of the ECC Processor

The exponentiations involved in the Schnorr protocol may be implemented using RSA or ECC. However, ECC involves much smaller bit lengths compared to RSA and is efficient in hardware. Hence we implement the Schnorr protocol using 192-bit ECC over prime fields which offers higher security compared with 1024-bit RSA. Highly efficient ECC and ECDSA implementations for constrained environments can be found in [115, 117]. However, in this work, we present two new designs which are optimized both for area and timing suited for integration with the standard JTAG.

We use the 192-bit NIST ECC curve P-192 and work in prime fields (F_p). The curve parameters used in our ECC implementation is as follows [59]:

p: The order of the prime field F_p.
a,b: The coefficients of the elliptic curve $y^2 = x^3 + ax + b$.
n: The (prime) order of the base point P.

h: The cofactor.
x, y: The x and y coordinates of P.
P-192: $p = 2^{192} - 2^{64} - 1$
a = -3, h = 1
b = 0x 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1
n = 0x FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831
x = 0x 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012
y = 0x 07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811

The point operations over affine and projective coordinates are performed by standard formula taken from the literature and are detailed in Section 4.2.9 and Section 4.2.10 respectively. In general, projective coordinates are introduced to avoid the relatively costlier inversions required in point-operation over affine coordinates. However, relativity among the costs of multiplication and inversion in F_p varies on their implementations. For example, when modular multiplication is computed in a bit serial fashion, it leads to log_2 p number of iterations (clock cycles); whereas, when inversion is performed by binary Euclidean algorithm, it requires at most 2 log_2 p number of iterations (clock cycles). Following the above design technique, the point operations over affine coordinates outperform projective coordinates.

4.2.9 Point Addition and Point Doubling in Affine Coordinates

Let P = (x_P,y_P) and Q = (x_Q,y_Q) be two points (not negative of each other) on the ECC curve in the finite field F_p.
The point addition of these two points can be represented as P + Q = R (x_R,y_R), where
$x_R = s^2$ - x_P - x_Q and y_R = -y_P + s(x_P - x_R)
s = (y_P - y_Q) / (x_P - x_Q)
Note that s is the slope of the line through P and Q.

Similarly, When y_P is not 0, then the doubling of point P is represented as 2P = R (x_R,y_R), where
$x_R = s_2$ - 2x_P and y_R = -y_P + s(x_P - x_R)
s = (3${x_P}^2$ + a) / (2y_P)
‘a’ is one of the parameters chosen with the elliptic curve and ‘s’ is the tangent on the point P.

4.2.10 Formulae used for ECC Point Addition and Doubling in Projective Coordinates

The Explicit formulae that is used in this thesis for implementing ECC Point Addition and Doubling in Projective Coordinates is shown in Table 4.1.

Table 4.1: Explicit Formulae for ECC Point Addition and Doubling in Projective Coordinates

Point Addition	Point Doubling
Cost	
12 Field Multiplications + 2 Squarings + 6 additions + 1 shift	7 Multiplications + 3 Squarings + 5 additions + 4 shifts + 1 cubing
Source	
1998 Cohen–Miyaji–Ono [30]	Efficient elliptic curve exponentiation using mixed coordinates - 2007 Bernstein–Lange [31]
Formulae	
Y1Z2 = Y1*Z2 X1Z2 = X1*Z2 Z1Z2 = Z1*Z2 u = Y2*Z1 - Y1Z2 uu = u*u v = X2*Z1 - X1Z2 vv = v*v vvv = v*vv R = vv*X1Z2 A = uu*Z1Z2 – vvv - 2*R X3 = v*A Y3 = u*(R - A) - vvv*Y1Z2 Z3 = vvv*Z1Z2	w = 3*(X1 - Z1)*(X1 + Z1) s = 2*Y1*Z1 ss = s*s sss = s*ss R = Y1*s RR = R*R B = 2*X1*R h = w*w - 2*B X3 = h*s Y3 = w*(B - h) - 2*RR Z3 = sss

Here ‘*’ indicates modular multiplication which in our case has been implemented using the Montgomery Multiplier. The addition and subtraction operations denoted here are all modular in nature. Using these set of formulae have the additional advantage that the computations are not dependent on the value of parameters ‘a’and ‘b’.

4.2.11 Hardware Implementation of space-time optimized ECC modules

4.2.11.1 Design I: ECC over Projective Coordinates

In this implementation, we use the 1998 Cohen-Miyaji-Ono mixed coordinates for point addition [30] and the 2007 Bernstein-Lange formulae [31] for point doubling from Explicit Formula database for Short Weierstrass curves [1]. To reduce area overhead, the adder and the Montgomery multiplier used in ECC have been optimized. The ordinary adder/subtractor (required for the intermediate operations of the Montgomery Multiplier) has been combined with the modular

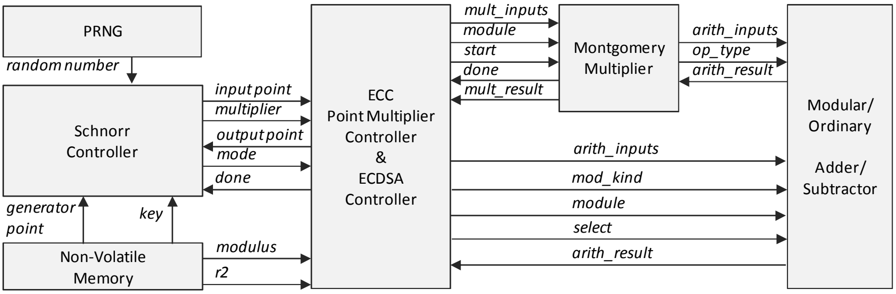


Figure 4.5: Block diagram of the security architecture for Design I

adder/subtractor (required for the modular addition/subtraction operations for implementing ECC in projective coordinates) using a 2-bit select signal. This helps reduce the area overhead further. The modules employed in our design are described below.

Schnorr and ECDSA Controller Modules: Figure 4.5 shows the block diagram of the Design I implementation. The Schnorr protocol consists of two main operations: ECC point multiplication and modular multiplication. In order to perform an ECC point multiplication, the Schnorr controller loads correctly the inputs of the ECC and ECDSA controller, and sets the 'mode' signal to '0'. Then it disables the 'reset' signal of the ECC and ECDSA controller so it can initiate the execution. Then the Montgomery multiplier and the modular/ordinary adder/subtractor blocks are used to perform the point multiplication. As can be seen, the adder/subtractor block is shared between the ECC controller and the Montgomery multiplier. If the ECC controller is using it, it sets the 'select' signal to '1' and then it chooses the operation type by setting the 'mod_{kind}' signal ('0' for addition and '1' for subtraction). Each ECC scalar multiplication is performed using the Montgomery Powering Ladder algorithm, which is also protected against Simple Power Analysis (SPA) attacks.

On the other hand, to perform a modular multiplication, we reuse the Montgomery multiplier. For that purpose we use the order of the prime number as modulus instead of the prime number itself. The Schnorr controller sets the mode to '1' and then uses the ECC controller as an interface to the Montgomery multiplier block. This interfacing was implemented in order to reuse the ECC controller finite-state-machine.

The ECDSA operation is performed partially by the ECC and ECDSA controller and partially by the Schnorr controller. It first executes all the ECDSA steps which require only integer multiplications by setting the mode to '2' and loading the signature into the ECDSA block. Then the Schnorr controller saves these intermediate values and reuses the ECC controller block to run the two final point multiplications. For executing the inversion present in the ECDSA protocol we used the Itoh-Tsujii algorithm [66] based on the Fermat's little theorem that allows to execute inversions using a modular multiplier.

We are reseeding the LFSR in Figure 4.5 after every authentication execution, with a new seed to avoid starting it with the same initial value on power up, in order to prevent replay attacks. Efficient LFSR reseeding techniques [136, 102, 129] using seed storage methods or seed derivation from the modules of the design can be used for the purpose. For security, the LFSR length must be large enough to prevent brute-force attacks (192-bit in our design) and irreducible polynomials used for the feedback taps to have all possible sequences (2^{192} as in our case). Moreover, the reseeding must be done quite often to prevent prediction of generated sequences (at the beginning of every authentication as in our case). The new seed value is loaded into the LFSR as soon as the 'request_unlock' signal in Figure 4.4 goes high. Though the proposed protocol works with pseudo-random numbers, for enhanced security, True Random Number Generators (TRNGs) based on Fibonacci or Galois Ring Oscillators [57], which have similar area overhead as LFSRs and substantially high randomness and unpredictability properties, can also be employed.

Montgomery Multiplier algorithm is the most common method for a fast implementation of modular multiplications and is also employed in our work. Carry Save Adders are used for the intermediate computations for addition/subtraction operations and then a full addition is performed to convert the final carry-save result into a conventional form. CSA adders have a small area and avoid carry propagation, i.e., are computed in constant time independently of the operands' length. However, a modular adder/subtractor is needed for ECC.Hence, we have modified the adder/subtractor block to have an ordinary addition/subtraction also, in order to use this block in our Montgomery multiplication implementation.

4.2.11.2 Design II: ECC over Affine Coordinates

The execution of the Schnorr protocol and ECDSA consists of several finite field operations (including inversion) and operations on elliptic curves. In Weierstrass elliptic curve, a point is primarily defined over Affine (x, y) coordinates which is further redefined over several Projective coordinates with the help of a third variable (X, Y, Z) in order to avoid inversion in point operations performed by chord-and-tangent method. A single inversion is eliminated by several (4-12) multiplications in Projective coordinates - still research is going on for finding coordinate systems to lower down multiplications in a point operation.

However, the implementation technique also plays an important role for improving efficiency of elliptic curve operations under a resource constrained environment like JTAG. It is already described in Section 4.2.8 that delay of a binary inversion/division method is just twice that of a bit-serial multiplication where both

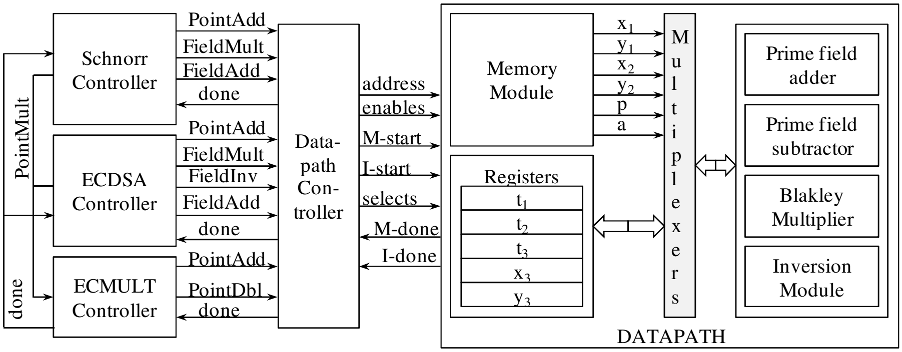


Figure 4.6: Block diagram of the security architecture for Design I

of them are assumed to be implemented by simple adder circuits - demands area in the same order. Thus design II attempts to implement a compact and flexible architecture for executing Schnorr protocol and ECDSA over Affine coordinates. Besides, this design computes all modular operations directly on 2's complement binary domain that avoids cost of domain conversions compared to the first design.

Flexible Datapath: In order to reduce the complexity of the controller logic, design II consists of a flexible datapath having all arithmetic blocks. There are two top level controllers in our secure JTAG implementation, namely ECDSA-controller and Schnorr-controller. These controllers generate instructions like PointAdd, PointMult, FieldAdd, FieldMult, FieldInv, FieldSub. In the next lower level, there is an ECMULT-controller which primarily generates two instructions - PointAdd and PointDbl. All instructions generated by top level controllers are first checked by the ECMULT-controller which further passes through the next lower level. Except PointMult, all other instructions are directly executed by the datapath shown in Figure 4.6. The instruction PointMult consists of PointAdd and PointDbl instructions which are generated in proper sequence by the ECMULT-controller. All controller logic in design II are realized as finite state-machines in which the final state sends a done signal to its predecessor. With the help of five temporary registers in the datapath shown in Figure 4.6, a point operation (point doubling or point addition) is computed as a single instruction - in which case the output of an execution is stored and supplied back to the memory through x3 and y3 ports. On the other hand, the outputs for all other finite field operations are directly generated from individual arithmetic units. In order to execute PointAdd instruction the datapath takes the input data from ‘x1’, ‘y1’, ‘x2’, ‘y2’, ‘p’, and ‘a’ ports, whereas for executing individual finite field operation, the ports are configured by the upper level controller logic.

Prime field Multiplication: In this design we use Blakley multiplication which is based on the iterative execution of doubling and addition. All internal operations are performed in respective prime field, that is intermediate results are always in their reduced form. Hence, the costly final reductions are eliminated. The multiplier unit contained in the datapath block (Figure 4.6) computes a modular multiplication in log_2 p number of clock cycles, assuming that both the operands also have lengths of log_2 p.

Prime field Inversion and Division: The prime field inversion and division could be efficiently computed by binary inversion division algorithm, which is based on binary Euclidean algorithm. The current design follows the implementation of such a unit that is described in [55]. The current module can compute one inversion (used in ECDSA) as well as one division (used to execute PointAdd and PointDbl) in $2log_2$ p number of clock cycles. Design II executes a PointAdd instruction in 5 log_2 p + 6 clock cycles and PointDbl in 4 log_2 p + 8 clock cycles. The clock cycles required to execute a PointMult instruction is: log_2 k * (4 log_2 p + 8) + (# k - 1) * (5 log_2 p + 6), where k represents the scalar multiplier in kP and # k indicates the Hamming weight of k.

4.2.12 Area and Timing Costs

We present here the area and timing results of our implementations. Both ASIC and FPGA results of the overall secure JTAG design with the sub-modules are mentioned.

4.2.12.1 Area Overhead

The ASIC area requirements in terms of gate equivalents (GEs) (synthesized with Synopsys Design Compiler v2009.06 for a Faraday 130 nm library) for the modules used in our ECC implementation are given in Table 4.2. The FPGA Synthesis results on Xilinx ISE 12.4 (with Virtex 6 xc6vlx75t family) for the modules are also presented.

The Modular multiplier in Table 4.2 is a Montgomery multiplier for Design I, while it is a Blakley multiplier for Design II. Hence, as shown in the Table, we require a total of 46716 GEs for Design I and 47050 GEs for Design II to implement the secure JTAG Scheme with the Schnorr and ECDSA controllers. We choose the solution described in [104] for having an estimate of area overhead of our approach. The cost of that solution is 25 k gates. It means that our solution is around twice larger than the solution in [104], which provides one-way authentication of the secure JTAG to the test server and and is based on symmetric-key cryptography. Our solution is based on public key cryptography, which inherently demands more hardware resources than symmetric-key based approaches.

Table 4.2: Hardware cost of secure JTAG

Module	Design I ASIC(GEs)	Design II ASIC(GEs)	Design I FPGA(GEs)	Design II FPGA(GEs)
Arithmetic unit (modular adder and subtractor)	1374	5128	164	311
Modular multiplier	5152	7314	615	756
Inversion module	-	24313	-	1482
Controller and data multiplexers	40190	10295	2189	531
Total	46716	47050	2968	3080

The area requirement in our designs can be reduced further by making use of a tiny custom microcontroller with an Instruction Set Extension (ISE), as in [64]. Here only the top-level ECDSA commands are managed with a processor. Moreover, replacing the Montgomery Multiplier (suitable for general prime-field operations) with more efficient multipliers employing Mersenne-like NIST prime reduction suitable for prime fields over F_p can also help reduce the execution time for an Elliptic Curve scalar multiplication.

There are of course much more compact implementations available in the literature, for instance, the 192-bit ECDSA implementation in [64] employing the same NIST recommended curve as in our case requires only 19.1 KGEs (thus consuming 23.5% less than the approach in [104]) and 859,188 cycles in total for the combined operations of ECDSA, Hash and pseudo-random number generation required for the protocol execution. Similarly the most area efficient 163-bit ECC implementation in [78] consumes only 12.5 KGEs (thus taking half the area in [104]) and 275,816 cycles for one Elliptic Curve scalar multiplication. In this work, though we did not achieve such high compactness, we have shown the feasibility of integration of the JTAG with the ECC and ECDSA modules by presenting combined area and timing results which have limited overheads.

4.2.12.2 Timing Overhead

The impact of the proposed solution in the use of the JTAG standard consists of an initial delay for executing the Schnorr protocol/ECDSA. Once the authentication and the signature verification steps are finished, the JTAG is unlocked and the JTAG instructions can be used without any timing overhead.

The initial delay is due to three main operations: 1) the time to request the unlock (associated with the time to insert the instruction ‘UNLOCK’) and the

Table 4.3: Detailed Timing Estimates

Scenario	Operation	#Clock cycles (Design I)	#Clock cycles (Design II)	Clock class
1. One way Authentication (DUT is the prover)	Unlocking	13	13	TC
	Time to shift data in and out	768	768	TC
	Protocol (1 k.P* operation)	3068150	240762	FC
2. One way Authentication (DUT is the verifier)	Unlocking	13	13	TC
	Time to shift data in and out	768	768	TC
	Protocol (2 k.P* operation)	6136692	482130	FC
3. Mutual (Two-way) Authentication	Unlocking	13	13	TC
	Time to shift data in and out	960	960	TC
	Protocol (3 k.P* operation each for prover and verifier)	9204842	722892	FC
4. ECDSA	Unlocking	13	13	TC
	Time to shift data in and out	576	576	TC
	Protocol (2 k.P* operation)	6137075	482324	FC

Legend: TC- Test Clock FC- Functional Clock

time to release the lock; 2) the time to shift in the protocol inputs and shift out the protocol outputs using the JTAG controller; and 3) the time to perform the protocol operations, including ECC point multiplications, ordinary multiplications and additional operations, to communicate between the dedicated Schnorr protocol modules. The first two operation types are measured in test clock cycles that depend on the JTAG frequency, while the last operation type is measured in functional clock cycles, the functional clock being usually faster than the test clock. The timing overhead is presented in Table 4.3, where we have distinct four scenarios.

The first scenario is a one-way authentication (manufacturer environment in Appendix B) in which the DUT acts as prover (A) and the test server acts as verifier (B). The only scalar (point) multiplication performed for A is n_a.P for generating T_a. The second scenario is a one-way authentication (manufacturer environment in Appendix B), but the roles of prover and verifier are reversed. Here the DUT acting as the verifier B performs two scalar multiplications (s.P and n_b.Pa). The third case is the two-way authentication (in the field update in Appendix B). Here, both A and B perform three scalar multiplications (n_a.P, s1.P and n_a.Pb for A, and n_b.P, s.P and n_b.Pa for B). Finally, the last one is the timing overhead associated with the execution of the ECDSA signature verification, which requires two scalar multiplications. In Table 4.3, 'k.P' indicates one Elliptic Curve Scalar Multiplication.

For having an estimation of time in milliseconds, we suppose a 100MHz clock frequency for the JTAG Test clock, and 115 MHz as functional clock frequency for Design I and 123 MHz for Design II, as shown in Table 4.4. The functional clock frequency is the maximum operating frequency obtained from FPGA synthesis. The test clock and the functional clock can be also the same without involving

Table 4.4: Time delay for authentication (ms)

	Functional Clock (MHz)	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Design I	115	26.67	53.37	80.05	53.37
Design II	123	1.96	3.93	5.89	3.94

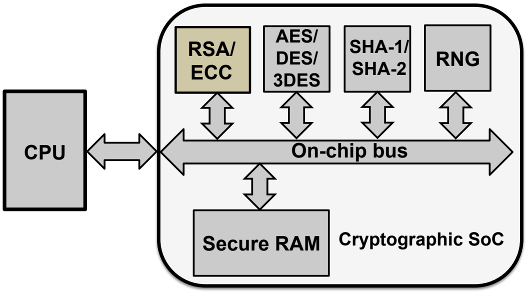
any design change. Considering the mutual authentication scenario with ECDSA signature verification, Design I has an initial delay of 133.42 ms while Design II has an initial delay of 9.83 ms.

4.3 Secure Test Wrapper

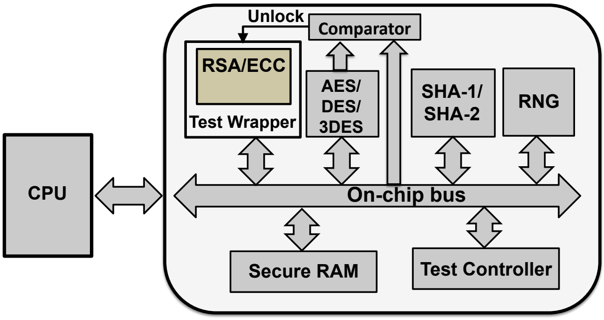
In this section, we present our proposed Secure Test Wrapper as a countermeasure at the system or SoC level. First, a brief introduction is provided to Cryptographic SoCs and the cryptographic SoC testing scenario. Then, we present our proposed countermeasure at the SoC level. The first approach is based on the reuse and modification of existing IEEE 1500 SoC testing standard. We present two flavors of these schemes which are based on Challenge-Response based protocols: KATAN based Secure Test Wrapper (STW) and PUF-based STW. Another countermeasure is based on a combined approach of Logic BIST reusing cryptographic modular multipliers and scan testing employing Secure Test Wrappers.

4.3.1 Cryptographic SoC

Security protocols combine many different cryptographic algorithms to establish and maintain a sufficiently secure and privacy preserving connection between two communication parties. A hardware platform intended to run such a protocol (e.g. on a smart card) must be able to implement public key cryptographic primitives such as RSA [114], ECC [74, 91] or El Gamal [89] and signature schemes (DSA [98], ECDSA [29]) in order to provide key establishment or signature generation and verification. In addition, it must also support symmetric-key primitives such as block ciphers (AES [39], DES [97], 3DES [101]) for implementing encryption/decryption operations and hash functions (SHA-1 [99], SHA-2 [100]). In order to reduce the bottleneck they create, these computationally intensive and time consuming algorithms are often realized as hardware co-processors. A typical architecture of a cryptographic SoC which supports different security protocols is depicted in Figure 4.7a.



(a) A Cryptographic SoC.



(b) Testing of Public-key Crypto Cores.

Figure 4.7: Cryptographic SoCs

4.3.2 Secure Testing and SoC Integration Testing Environment

Today as a result of globalization, the development and fabrication of advanced integrated circuits (ICs) is typically migrating offshore. This migration to third-party providers and to low-cost foundries has made ICs vulnerable to security compromise, functional changes, information leaks or even system failures under specific conditions. Such intrusions may pose a major threat to embedded systems in critical applications and infrastructures. These risks have been considered not only in the academic community but also in the fabless semiconductor industry and governmental agencies. In this context, secure testing environments are becoming more important in ensuring a trustworthy hardware environment.

System-on-Chip (SoC) integrators often use embedded cores which may be procured externally from various IP vendors. Standard functional testing or Design for Testability (DfT) methods cannot be employed directly for IP testing in SoCs. The security of an SoC depends on the resistance of these IP cores to attacks exploiting the existing on-chip DfT. Particularly, cryptographic IP cores must be protected from such attacks. Due to the possible attack scenarios on SoCs, the customers

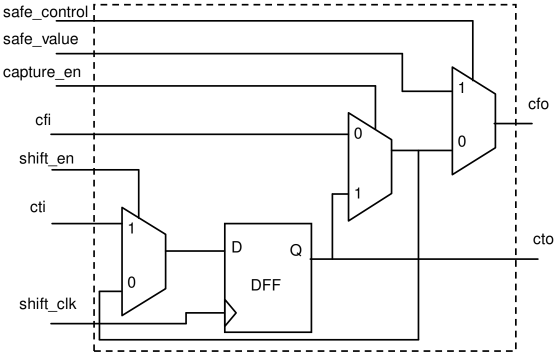


Figure 4.8: Standard IEEE 1500 Test Wrapper Cell

need to securely test the individual cryptographic IP blocks after the deployment of SoCs.

There have been many approaches at secure DFT, as mentioned in Section 3.8. However, these secure testing methods involve changes in the existing on-chip DfT or work only with specific cryptographic implementations. Hence, they may not be suitable for SoC integration where individual IP blocks needs to be included in the SoC generically, without internal modifications.

4.3.3 Standard IEEE 1500 Test Wrapper

IEEE 1500 Test Wrapper [8] has emerged as the test standard for industrial SoCs. The standard Test Wrapper boundary cell, one type of which is shown in Figure 4.8, is used to provide both test access and test isolation to the core and the external user-defined logic during testing. The core test input (cti) is the test input to the wrapper cell. It can come from either a primary input (if the cell is the first cell in the wrapper chain) or the cto signal of the previous wrapper cell in the chain. The core test output (cto) is the test output of the wrapper cell. It can drive either a primary output (if the cell is the last cell in the wrapper chain) or the cti signal of the next cell in the wrapper chain. The core functional input (cfi) is fed from the user-defined logic for input wrapper cells. For output wrapper cells, this input is fed from the core. The core functional output (cfo) for input wrapper cells drives the core. For output wrapper cells, this output drives the user-defined logic [10].

The Test Wrappers have three modes of operation. In the INTEST mode, input vectors are applied to the core and the core response observed at the output. In the EXTEST mode, the user-defined logic surrounding the core is tested while the core itself is isolated. The NORMAL mode is the functional mode of operation.

4.3.4 Previous Work

The proposed design is based on the Secure Test Wrapper first introduced in [26]. However, the scheme had some problems. It was insecure, as it used LFSRs (with secret polynomial) for generating the key to unlock wrapper, and sent the test key in plaintext to the chip for comparison. LFSR length and polynomial was kept secret (security by obscurity) and even non-irreducible polynomials of LFSRs were used to increase resistance to brute-force attacks. It also modified the standard IEEE 1500 test wrapper boundary cell introducing additional flip-flops, increasing the test area overhead.

The scheme presented in this thesis preserves the existing structure of Standard Test Wrapper boundary cell. It is secure and does not send the key in plaintext. It also uses standard scan chains for thorough testing of the crypto cores and has limited area overhead. In this manner, it seeks to address the tradeoff between security, testability and test overhead. The design is highly scalable as it does not affect the existing scan-chain DfT present within the SoC and only adds some modules.

4.3.5 Katan-Based Secure Test Wrapper (STW)

4.3.5.1 Challenge Response Protocol

The proposed secure test architecture, shown in Figure 4.9, is based on a challenge-response based test protocol using the KATAN [25] light-weight block-cipher. The latter is a family of hardware-oriented block ciphers, suitable for constrained environments, as in secure smart cards or RFID chips.

KATAN runs in software on a secure server. The fixed key should also be securely stored on the server. KATAN receives its plaintext input, via a serial interface, in the form of a random number nonce RN, which is generated by the on-chip True Random Number Generator (TRNG). This output is also transmitted to the on-chip KATAN hardware implementation fed by the same secret key. The key is loaded from a secure on-chip non-volatile memory location by an external interrupt at the start of the testing process.

The ciphertext outputs generated by the software, $E^*_K(\text{RN})$, returned through the serial interface, and hardware, $E_K(\text{RN})$, are then evaluated in an on-chip comparator. When they match, only then an ‘Unlock Wrapper’ signal unlocks the wrapper to enable normal testing of the cryptographic core using the scan chains. This unlock signal is connected to the enabling AND gates between the input and output wrapper boundary registers (IWBR and OWBR) and the AES core scan

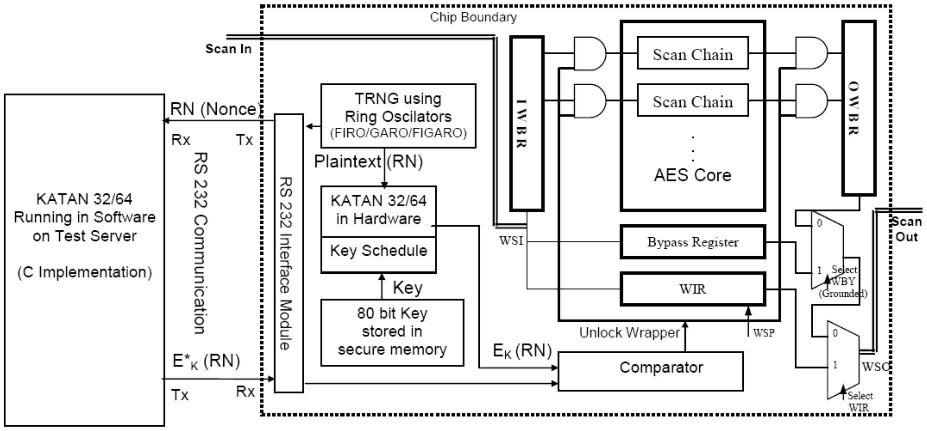


Figure 4.9: KATAN-based Secure Test Wrapper architecture

chains. The AES core is the circuit under test in this case. The test protocol involved in the design is presented in Figure 4.10.

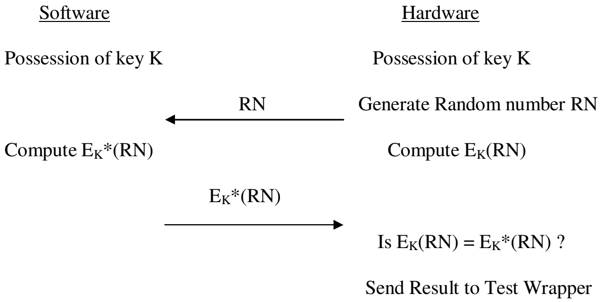


Figure 4.10: Secure Test Wrapper Protocol

The above protocol can only be executed by the SoC Integrator in possession of a valid key. The scan chains are otherwise locked against attacks by disabling the scan-in and scan-out. After successful entity authentication of the secure chip, all communication between the test server and the secure chip is encrypted using the secret key to prevent an attacker from hijacking the communication after authentication.

A brief description of the main building blocks used in the secure test design is now provided.

4.3.5.2 KATAN Lightweight Block Cipher

In KATAN (shown in Figure 4.11), the plaintext is loaded in two registers L1 and L2. In each round, several bits are taken from the registers and entered in two nonlinear Boolean functions. The output of the Boolean functions is loaded to the least significant bits of the registers (after they are shifted). This is done in an invertible manner. To ensure sufficient mixing, 254 rounds of the cipher are executed.

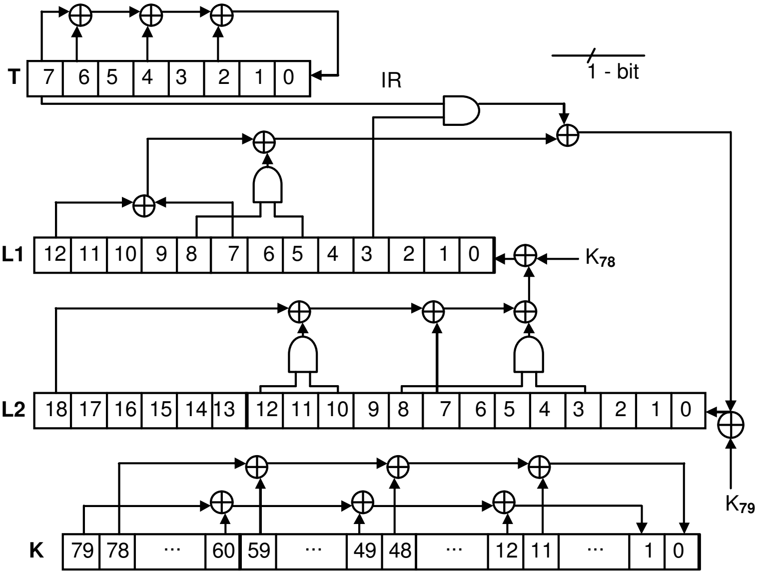


Figure 4.11: KATAN 32 Round Function

An LFSR T is used instead of a counter, for counting the rounds and to stop the encryption after 254 rounds. As there are 254 rounds, an 8-bit LFSR with a sparse feedback polynomial can be used. The LFSR is initialized with some state, and the cipher has to stop running the moment the LFSR arrives at the predetermined state.

The key schedule of the KATAN cipher loads the 80-bit key into an LFSR K (the least significant bit of the key is loaded to position 0 of the LFSR). In each round, positions 79 and 78 of the LFSR are generated as the round's subkey $k_{2.i}$ and $k_{2.i+1}$, and the LFSR is clocked twice. The feedback polynomial that was chosen is a primitive polynomial, with minimal hamming weight of 5: $x^{80} + x^{61} + x^{50} + x^{13} + 1$. In other words, let the key be K, then the subkey of round i is $k_a \parallel k_b = k_{2.i} \parallel k_{2.i+1}$,

where $k_i = K_i$ for i = 0 . . . 79
$= k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13}$, otherwise [25].

4.3.5.3 True Random Number Generators (TRNGs) using Ring Oscillators

TRNGs based on the approach by Golic [57], provide a truly random sequence of bits, as compared to pseudo-randomness obtained from Linear Feedback Shift Registers (LFSRs). LFSRs generate a predictable sequence of bits due to their linearity property and it is possible to deduce the next sequence once the seed and feedback polynomial are known.

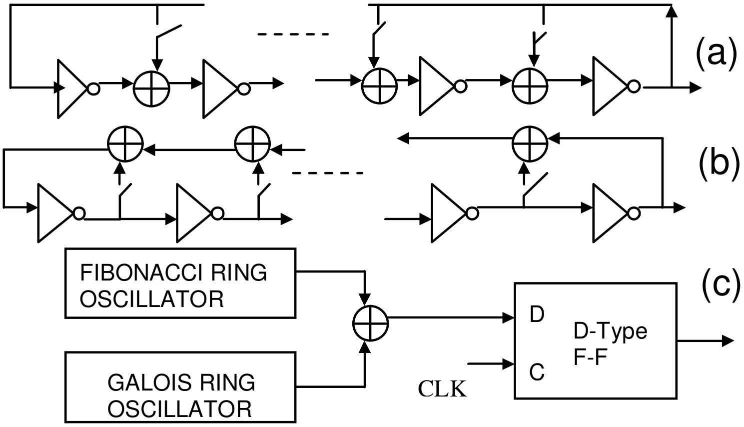


Figure 4.12: True Random Number Generators (TRNGs) (a) Fibonacci Ring Oscillator; (b) Galois Ring Oscillator; (c) TRNG using combination of both

TRNGs, as shown in Figure 4.12, are composed of an odd-number of inverters with XOR gates in the forward or feedback paths, for the Fibonacci and Galois Ring Oscillators respectively. To enhance randomness, the outputs of two types of ring oscillators are combined together. The frequency of the ring oscillators depends on the number of inverters in the chain, the feedback used, and the fabrication process technology employed. There should be a mechanism to take the output from the TRNG only after sufficient number of clock cycles have elapsed, as the output of TRNGs is not completely random at the beginning and randomness increases over time. In the proposed implementation, there is a concurrent process which keeps checking the count of elapsed clock pulses, and only after a predetermined number of clock cycles, the output of the TRNG is fed into an 32 or 64-bit register (for KATAN 32 or KATAN 64) to be used as the plaintext random number nonce in the challenge-response protocol.

4.3.5.4 Security Analysis of the Test Protocol

Our analysis is based on some feasible security assumptions. It is assumed that the attacker does not have access to the Test Server or the key used to run KATAN in software. The attacker cannot probe the 'Unlock Wrapper' signal on the chip, otherwise he can just flip the bit (for instance, by pointing laser light) to unlock the wrapper in order to enable the scan chains. This can be achieved by laying the signal line in the deep metal layers of the SoC during fabrication to prevent any kind of probing attacks on it. Another assumption is that the attacker cannot observe the bit patterns of the on-chip $E_K(\mathrm{RN})$ signal. Otherwise, he can just send $E_K^*(\mathrm{RN})$ matching with this sequence, to unlock the wrapper. This can also be achieved by a similar approach as in the last case. It is also assumed that the TRNG is secure against any types of fault attacks (such as lowering of temperature) and cannot be forced to generate the same nonce (such as all zeroes).

The KATAN Block Cipher is assumed to be secure. This can also be taken as a safe assumption, as there are no known practical attack against this block cipher. Differential fault attacks on KATAN appear in [123] and in [14], however the attack complexity of 2^{59} is too high to be considered practical for a lightweight cipher. KATAN can also be replaced by other lightweight block ciphers, such as PRESENT, for implementing the entity authentication mechanism.

Some attack scenarios are considered. Chosen plaintext attack is not feasible as the plaintext is a true random number nonce. Replay attacks are not useful for an attacker, due to the same reason. The attacker can only eavesdrop on the serial communication taking place between the secure chip and the test server. Since the random number nonce is generated on-chip and sent serially to the server, while the ciphertext on this nonce is generated by the server, it makes no sense to replay it back, as each time, a new nonce is generated by the TRNG. Man-in-the-middle attack is feasible only if the attacker has physical access to the chip as well as the communication taking place between the secure test server and the chip.

The probability of mounting successful attacks on the proposed secure test structure is now stated briefly. The probability of collision in a Guessing attack (or Brute Force Attack) is 2^{-64}, if 64 bits is the block length (KATAN 64) or 2^{-32}, if 32 bits is the block length (KATAN 32). Attacks on the block cipher have the same probability. The complete security analysis of the KATAN block cipher can be found in [25].

4.3.5.5 Key Distribution Problem and Possible Solutions

The key to unlock the wrapper should be securely transmitted from the core providers to the SoC integrators. We propose two main solutions for this problem: (1) the key is transferred securely through some communication channel, or (2) the key is stored inside the core itself in such a way that only authorized SoC integrators can access it. One simple mechanism for solution (1) could consist of establishing a Public-Key Infrastructure (PKI) between integrators and providers and sending a list of part numbers and corresponding keys to the integrators, encrypted using their public key. Embedding the keys in solution (2) can be done using the same PKI, but by storing the encrypted key on the chip itself. Note that this does not require the chip to perform any public-key encryption; it merely stores the encrypted form of the key. The enhanced protocol can be stated in Figure 4.13.

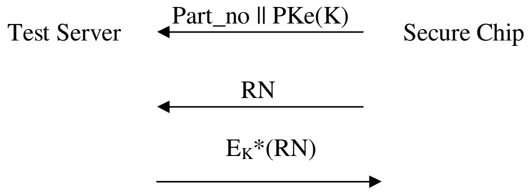


Figure 4.13: Modified Secure Test Wrapper Protocol

Here, PKe(K) refers to the key K encrypted with the public key of the SoC Integrator, stored on-chip by the core provider. The key K is only accessible by the authorized SoC integrator using their corresponding private key. However, the chip must now store both K and PKe(K). Elliptic Curve Cryptography (ECC) can be used to reduce storage requirements.

4.3.5.6 Hardware Implementation

Synopsys DfT Compiler Version C - 2009.06-SP3 with 130nm Faraday library, is used for standard scan chain insertion into the AES crypto-core (which is used as a test case), followed by standard IEEE 1500 Test Wrapper Insertion by the same tool. Then, the entire design with the other test modules - TRNG, KATAN hardware implementation, on-chip key memory, comparator and the RS-232 serial interfacing module, is compiled in Synopsys Design Compiler.

To reduce the test area overhead in the stardard IEEE 1500 Test Wrapper insertion on large cryptographic circuits, the wrapper boundary registers are reused for the implementation of the KATAN Block cipher, the TRNG storage and the other secure test modules. Scan flip flops immediately after (before) a core input have been reused as wrapper input (output) cells, making additional dedicated test wrapper cells unnecessary [87].

FPGA Prototyping has been done to verify the correct operation of the Design. The TRNG has been implemented using Look-Up Tables (LUTs) and Controlled

Place constraints in Xilinx ISE and mapped onto a Spartan 3E FPGA (xc3s500E). The randomness property of the sequences generated by the TRNG has been tested using standard NIST and Diehard Randomness Tests.

In our test approach, we have used a RS-232 interface for serial communication between the secure chip and the test server to reduce the test area overhead. However, a JTAG interface with an IEEE 1500 standard Test Access Mechanism (TAM) may also be employed for other suitable applications.

4.3.5.7 Area Results

The ASIC area comparison for the proposed secure test design, including the serial interface, in Synopsys Design Compiler synthesis is shown in Table 4.5. Implementation of the AES core requires 9317 gates and 263 flip-flops (FFs). The AES core, with 10 and 20 scan chains, requires similar additions in the number of gates. The secure test wrapper on the 10 scan chains results in an area overhead of 9.72% over the Standard IEEE Test Wrapper insertion. The design with 20 scan chains, on the other hand, requires only slightly less gates than the one with 10 scan chains, incurring an area overhead of 8.48% over the standard test wrapper. A custom Perl script can be used for optimized wrapper insertion using the method proposed in [87], instead of the standard insertion by the Synopsys DfT Compiler to further reduce test area overhead.

The scan registers are reused for the construction of the KATAN block cipher, the storage of random number generated by the TRNG, and other modules in the proposed design, to reduce area overhead. To implement KATAN (Figure 4.11), we require 120 FFs: 8 FFs for the round count register T, 13 FFs for the shift register L1, 19 FFs for the shift register L2, and 80 FFs for the key register K. Part of the 263 Scan FFs is reused for the purpose. The fact that the random number generation and scan test does not occur simultaneously permits this reuse. However, a few additional multiplexers are required to switch between the normal scan and the random number generation modes.

As seen in the Table 4.5, most of the area overhead in the proposed secure test design comes from the test wrapper architecture which is emerging as the leading test standard for future SoC core testing [88]. This test approach is the first of its kind to use a cryptographic primitive to provide both security and testability, preserving the complete scan chain structures.

4.3.6 PUF-Based Secure Test Wrapper

In Section 4.3.5, the high testability of scan chains has been combined with a secure Test Wrapper (STW) to achieve restricted access with reasonable test area overhead.

Table 4.5: Area comparison (Number of Gates Required)

Design	**10 scan chains**	**20 scan chains**
AES Implementation	9317	9317
AES with Normal Scan	9725	9666
AES with Scan and Standard Test Wrapper	11415	11309
AES with Scan & Secure Test Wrapper	12525	12268
Test Area overhead of Secure Test Wrapper over Std. Test Wrapper	9.72%	8.48%

This has been achieved using a light-weight block cipher to activate an unlocking mechanism for the test wrapper, to allow direct access to cryptographic IP blocks on the tester equipment by scan chains [42]. The authentication mechanism with the block cipher requires additional non-volatile memory (NVM) to store the secret key, which may be susceptible to side-channel analysis.

In this part of the thesis, we propose a PUF-based secure test wrapper which provides a secure test environment allowing only eligible testers to test the individual IP blocks in a SoC. The requirement of secure-key storage on NVM is overcome by using a Hamming distance based challenge-response authentication mechanism using physically unclonable functions (PUFs).

4.3.6.1 Physically Unclonable Functions (PUFs)

PUFs are promising security primitives that exploit the physical characteristics of a device. PUFs can be used as secure key storage [127, 79] and in authentication protocols [125, 50]. A stimulus applied to a PUF is called challenge C, and the reaction of the PUF is called response R. The response depends on both the challenge and the unique intrinsic randomness of the device. Since PUFs are based on the physical properties of the device in which they are embedded, no other entity can verify the response of a PUF to a given challenge without a priori knowledge of an authentic Challenge Response Pair (CRP). A comprehensive overview on PUFs can be found in [83].

Typical assumptions on PUFs are [83, 84]:

- *Unpredictability:* An adversary cannot predict the response to a challenge of a specific PUF. Moreover, the response R_i of one CRP (C_i,R_i) gives only a negligible amount of information on the response R_j of another CRP (C_j,R_j) with $i \neq j$.

- *Unclonability:* An adversary cannot emulate the behavior of a PUF on another device since the behavior is fully dependent on the physical properties of the original device.
- *Tamper-evidence:* Any physical attack against a PUF will irreversibly and randomly change its challenge-response behavior.
- *Robustness:* For a single PUF, the difference between two responses of a particular challenge should be small.

Among different PUF architectures, we focus on silicon PUFs, which can be easily applied to ICs. In this work, we employed Arbiter PUFs (APUFs) [80] due to the possibility of fast prototyping on FPGAs. However, ring oscillator based PUFs, which can be efficiently implemented on FPGAs, could have been used as well. APUFs are based on race conditions on two identical logic paths. Independent of the semiconductor manufacturing technology, there are inevitable variations in the propagation delays of logic paths. If one implements two identical logic paths which are controlled by a challenge and get triggered simultaneously, due to different propagation delays, one transition occurs first. A digital arbiter captures this transition and indicates which of the two signals was the fastest and therefore produces a one-bit response. By using a challengeable logic path, the number of responses of a single APUF can be made exponentially large in the dimensions of the APUF (large CRP source), which makes them a good candidate to be used in authentication mechanisms. However, it was claimed in [116] that APUFs are subject to model building attacks that allow predicting responses with non-negligible probability. Further, the response of APUF cannot be used directly as a key in an authentication mechanism without post-processing, since APUF responses are obtained through measurements which suffer from inevitable environmental noise. Two queries of the same challenge may give different responses and eventually the authentication mechanism will not work properly. In order to counter these problems, additional primitives must be used: *controlled PUFs* [54] use cryptographic hardware to hide the responses of the underlying PUF from an attacker and *fuzzy extractors* (FE) [45] correct the errors in PUF responses.

4.3.6.2 PUF-Based STW

We propose an alternative secure test wrapper activation using a PUF-based authentication mechanism. This mechanism avoids key management of secret keys between parties, and allows only eligible testers to use the STW by using a CRP database of the underlying PUF. While this mechanism prevents scan-based side-channel attacks, it allows the tester to individually test IP-blocks.

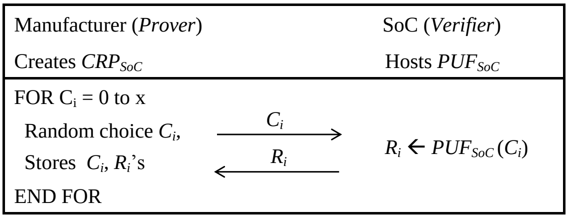


Figure 4.14: Enrollment of CRP database

4.3.6.3 Trust Model and Assumptions

As in most PUF based authentication schemes, we assume that the adversary can eavesdrop the communication channel and manipulate the messages exchanged with the SoC. However, similar to the PUF-based key storage [80], we assume that the adversary cannot access the response of the PUF. In our trust model, we are considering a SoC integration scenario; where the manufacturer is assumed to be trusted and the adversary cannot access the CRP database of the underlying PUF. In addition the internal communication within the SoC is assumed to be secure.

4.3.6.4 Secure Test Wrapper Activation Mechanism

Every PUF-based authentication mechanism requires an enrollment phase in order to create a CRP database CRP_{SOC}. Figure 4.14 shows the enrollment mechanism. The manufacturer queries randomly chosen challenges depending on the CRP space size (0 to x) and stores the challenge-response pairs (C_i, R_i). In normal mode, our design prohibits listening to *response* values in order to prevent model-building attacks [116]. Therefore, after creation of CRP_{SOC}, the manufacturer is able to create the database, and then disables read-out of PUF responses, e.g. by blowing fuses.

To prevent replay attacks, we propose a new mechanism expecting two challenges whose responses have a specific Hamming-distance. Figure 4.15 presents the authentication mechanism: First, the test server starts the protocol by sending a synchronization “SYN” signal to activate the LFSR which will create the pseudo-random number $\Delta \neq 0$. This Δ value refers to the Hamming distance between two responses of the intrinsic PUF of the SoC. At this point, the server will check the database and find two responses (R_i, R_j) which have Hamming distance “Δ”. Accordingly, (C_i, C_j) are sent by the test server to the SoC in order to make sure that the tester owns the right to enable the testing environment of an IP block on the SoC. If the SoC PUF generates responses within few errors than can be corrected, then the IP block will be enabled for testing using the scan chains. Otherwise, the mechanism fails and does not allow the test server to test the IP

Test Server (*Prover*) Stores CRP_{SoC}		SoC (*Verifier*) Stores PUF_{SoC}
Synchronize	$\xrightarrow{SYN}$	Activate PRNG
Receive "Δ" value	$\xleftarrow{\Delta}$	Set "Δ" value
Find $\{C_i$ and $C_j\}$,		$R_i \leftarrow PUF_{SoC}(C_i)$
with $\{\Delta = R_i \oplus R_j\}$	$\xrightarrow{C_i, C_j}$	$R_j \leftarrow PUF_{SoC}(C_j)$
Start testing	$\xleftarrow{ACK}$	*iff* $R_i \oplus R_j = \Delta$

Figure 4.15: PUF-based Secure Test Wrapper Protocol

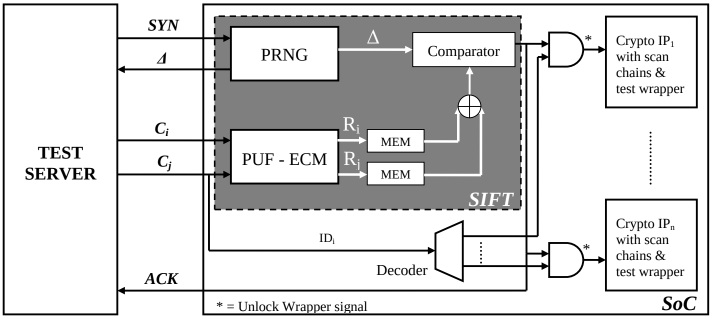


Figure 4.16: PUF-based Secure Test Wrapper Model

block. Finally, SoC sends an acknowledgment signal "ACK" back to the test server to indicate the authentication status.

4.3.6.5 Security of the Mechanism

The security of the scheme depends on the ability of the adversary to predict the response of the PUF values by observing the communication taking place between the SoC and the test server. The adversary can make a list of (C_i, C_j, Δ) pairs to mount a replay attack or to predict (C'_i, C'_j) for an unknown Δ'. As long as Δ is selected large enough and because of unpredictability of PUF, the probability of a successful prediction is negligible.

In order to prevent reuse of the same Δ, a mechanism reseeds the PRNG (LFSR) after every authentication process. Specifically for this design, we recommend the use of 32-bit LFSR module and 64-bit PUF challenges.

Table 4.6: Area comparison of the two secure test wrappers (STW)

Design	**KATAN based**	**PUF based**
AES Implementation	17495	17495
With Normal Scan	18985	18985
With Scan and Std. Test Wrapper	23934	23934
With Scan and STW	28932	26988
Overhead over Std. Test Wrapper	20.88%	12.76%

4.3.6.6 Hardware Implementation

The implementation of this authentication mechanism is given in Figure 4.16. The Secure Infrastructure for Test (SIFT) includes a PUF followed by an error-correction module [45], two registers (MEM) to store corresponding responses, XOR gates and an LFSR. The error-correction module (ECM) is selected as a simple majority voting mechanism for 11-bits. In this way we corrected the errors of the PUF up to 5-bits. A 32-bit LFSR is implemented to generate Δ, and a 64-bit LFSR is implemented for session challenges of the PUF.

4.3.6.7 Area Results

We compared the hardware area overhead of our PUF-based STW with a KATAN-64 based STW (in Section 4.3.5), which provides equivalent security to the 64-bit length PUF challenges used in our design. The results are shown in Table 4.6. All area results are provided in gate equivalents (GE) of 2-input NAND gates for a Faraday 130 nm library, synthesized in Synopsys' Design Compiler v2009.06, in conjunction with Synopsys' DFT Compiler. The results show that the PUF-based STW is more efficient in terms of area requirements than the KATAN block cipher based implementation.

4.4 Secure Built-In Self Test

Many of the security problems with testing cryptographic SoCs can be overcome by using a Built-In Self-Test (BIST). This approach eliminates or reduces the need for an external tester by integrating active test infrastructure onto the chip. Furthermore, it provides field-test capability which is essential for cryptographic cores and the internal registers of the device cannot be accessed from the BIST I/O interface. Hence, it prevents an adversary from revealing the secret data. However, BIST has relatively high area overhead and provides a good test quality only for a limited types of logic. Moreover, it provides only a pass/fail signature which is not useful for diagnosis.

Scan chains, on the other hand, provide the highest testability employing a more general and flexible approach. Additionally, the IEEE 1500 Test Wrapper provides a standard for wrapping multi-vendor IP cores in a SoC environment [8]. It is used to provide both test access and test isolation to the core and the external user-defined logic during testing. However, the fact that scan chains provide observability of the internal nodes from the I/O pins of a device, can be exploited to mount several scan-based attacks. In order to address this problem, there is ongoing research in developing a mechanism that ensures that only authenticated users can have access to the test interface using secure test wrappers (STWs) [42, 43].

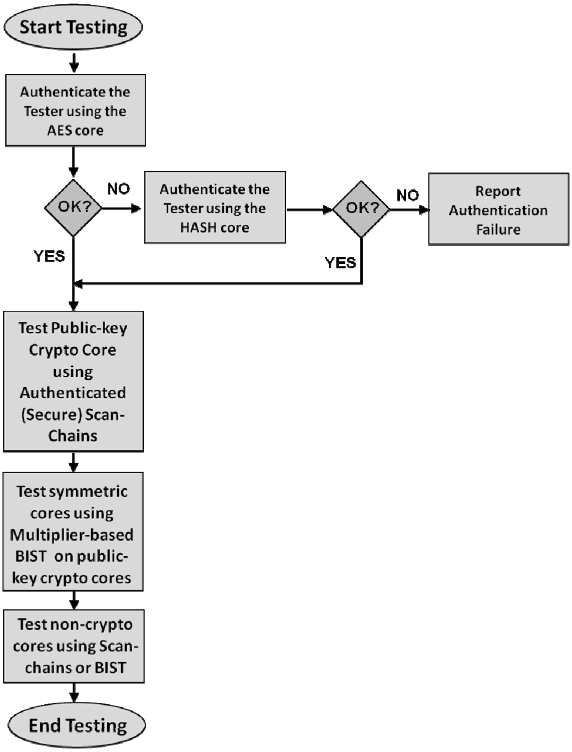
In this part of the thesis, we combine the two testing approaches – BIST and secure test wrappers and develop a modular and secure testing strategy for cryptographic SoCs, i.e SoCs that are capable of running cryptographic protocols. Symmetric cryptographic cores, such as block ciphers and hash functions, expose very good random pattern testability properties and can be very efficiently self-tested using BIST [119]. In order to minimize the introduced area overhead, we do not implement standard BIST infrastructure, but re-use a modular multiplier from an existing public-key core (RSA, ECC) and build BIST circuitry based on it. In order to test the public-key core itself and the other non-cryptographic modules in the SoC, e.g. a microcontroller, we use a secure scan-chain method employing the STW. Instead of adding a dedicated block cipher to implement the authentication framework [42], we re-use the one that is already available in a cryptographic SoC. In this way, we additionally reduce the area overhead while still preserving the test quality and security of a device.

4.4.1 Overall Strategy

The overall test strategy is depicted in the flowchart in Figure 4.17. The test process starts with the execution of the authentication protocol using the AES block cipher in order to check whether the testing server is genuine. Details about the authentication protocol and its execution can be found in Section 4.3.5 and [42].

Only if the authentication succeeds, the test wrapper around the scan-chains of the public-key module is unlocked for testing. Since it can be the case that the authentication protocol fails due to faults in the AES block itself, we foresee the possibility to run it using the hash function block. Only if the authentication fails in that case as well, the testing process is aborted and the failure is reported to

Figure 4.17: Secure and Efficient Testing Strategy Flow Diagram



the server. Otherwise, the testing continues. Section 4.4.5 explains the fail-safe scenario in more detail.

After proving that the test server is authentic, testing of the public-key core is performed using the secure test wrapper. Then, the multiplier from the tested core is re-used to perform the BIST of the symmetric-key cryptographic cores, i.e. the AES and the hash module. Note that in the case that the initial authentication has failed because of the errors in the AES core, its testing would not be necessary since the outcome of the pass-fail BIST would be already known. However, at this testing stage it is not known whether the authentication was executed using the AES or the hash module and hence testing of both modules has to be performed.

When the testing of security-critical cores is finished, the other non-cryptographic modules are tested. Depending on their architecture, either scan-chains or BIST approach could be used. Finally, the remaining module of the SoC in Fig. 4.7a: the RNG needs to be tested. It is performed by examining the randomness of its output, using online NIST or DIEHARD tests.

The secure testing of the different modules is now explained in more detail.

4.4.2 Testing of Public-Key Modules

The public-key cryptographic processor can be thoroughly tested using scan chain DfT. A security enhanced IEEE 1500 Test Wrapper is used in the form of a locking/unlocking mechanism to disable/enable access to the scan-chains. This can be performed using either the method presented in Section 4.3.5 or in Section 4.3.6.

The component under test (CUT) first sends a challenge to the test server. Using the a priori established key, the server encrypts the received challenge using the AES algorithm and sends the ciphertext back to the CUT. Using the same key stored in a non-volatile memory, the CUT calculates the same encryption using the existing AES core. When the received and calculated encryption results match, only then an Unlock Wrapper signal unlocks the wrapper to enable normal testing of the public-key core using scan chains, through the wrapper. This Unlock signal is connected to the enabling gates between the input and output wrapper boundary registers (IWBR and OWBR) and the public-key core scan chains.

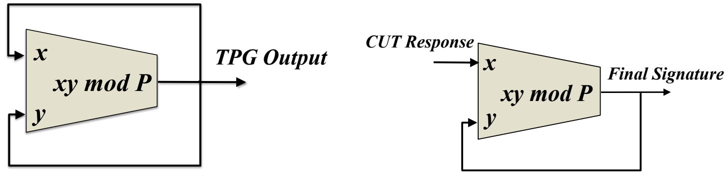
The AES block on the cryptographic SoC has been used in this entity authentication mechanism in the form of a challenge-response protocol (CRP) to allow access to the internal scan chains of the public-key processor only to eligible testers. This requires an on-chip key storage and the AES running on the Test Sever. The data communication between the SoC and the Test Server happens over a serial interface (for instance RS232) or the IEEE 1149.1 compatible JTAG port.

4.4.3 Testing of Symmetric-Key Modules

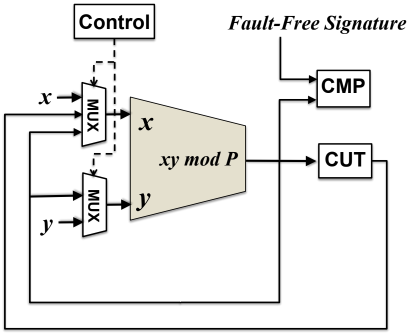
The symmetric-key modules in a SoC from Fig. 4.7a, i.e., block ciphers and hash functions, are tested using a pseudo-random Built-In Self-Test (BIST). Specific structures and operations used for the implementation of these modules provide high pseudo-random testability [119], which makes pseudo-random BIST a suitable test solution for this purpose. Rather than adding the standard BIST circuitry, we reuse an existing modular multiplier, which is an inherent part of a public-key module, to build the BIST infrastructure. It is shown that such a solution introduces less hardware overhead [72].

The standard BIST circuitry consists of a Test Pattern Generator (TPG), a Signature Analyzer (SA), and a BIST control unit (BCU). The TPG is used to produce test patterns which are fed into the Circuit Under Test (CUT) as a test stimulus. The SA compacts the CUT's responses into a single signature, which is then compared to the precomputed golden signature. The BCU is needed to activate the test and check the responses. If the two signatures match, the CUT is assumed to be fault-free, otherwise, an error is reported. Efficient test pattern generators and response compactors reusing the arithmetic functions shared with the main datapath have been proposed earlier in [124, 109, 108].

In what follows, we show how a modular multiplier can be configured to act as a TPG and a SA and explain how the presented testing strategy is applied to a cryptographic SoC. More detailed description of this testing strategy can be found in [72].



(a) Modular Multiplier as a TPG. (b) Modular Multipler as a signature analyzer.



(c) Configurable Modular Multiplier as both TPG and SA .

Figure 4.18: Multiplier based BIST strategy

- **Multiplier as a TPG:** Figure 4.18a depicts a modular multiplier configured to act as a TPG. By choosing a proper modulus P, such a configuration implements the *Blum Blum Shub* (BBS) algorithm [21] and provides a sequence of pseudo-random numbers at its output.
- **Multiplier as an SA.** When configured as shown in Figure 4.18b, the multiplier acts as a signature analyzer. It collects the CUT's responses on its input and provides the final signature at the output. [72] evaluates and proves the good quality of a multiplier based signature analyzer.
- **Configurable Multiplier as a TPG/SA:** By adding two multiplexers at its inputs and a simple controller, a multiplier becomes configurable to act

both as a TPG and SA (Figure 4.18c). The introduced hardware overhead of such an architecture is smaller than if a *regular* BIST infrastructure would be added to a design.

- **Testing Scenario.** To make understanding easier, we hereby explain a possible test scenario in a cryptographic SoC (Fig. 4.7b). A random pattern testable cryptographic core, such as a block cipher or a hash function, acts like a component under test (CUT). The modular multiplier from a public-key cryptographic core is re-used to produce the test patterns, collect the CUT's responses and create a digest of the final signature. The BCU controls the testing process and compares the obtained final signature with the precomputed golden one. The existing communication infrastructure in a SoC is used for data transfer between the modules involved in the testing process.

4.4.4 Testing of Other Non-Cryptographic Modules

Depending on their testability properties, the non-cryptographic modules can be tested either using BIST or the IEEE 1500 Test Wrapper. In case there is no security requirement for these modules, no activation mechanism would be required. However, in case the non-cryptographic modules need to be protected (for instance, for IP protection), they can also be subjected to the same authentication process using the multiplier-based BIST or Secure Test Wrappers.

4.4.5 Test Fail-Safe Scenario

If the AES block that initiates the authentication mechanism has some errors, the authentication mechanism fails. As a backup variant, the available hash module would serve to run the authentication protocol required for enabling the Test Wrapper around the public-key module, which would then test rest of the modules (Fig. 4.17). It is done by employing the available hash core in a *keyed-hash* mode, using the same shared key as in the case of the AES-based authentication. The only difference is that instead of the encryption of the generated challenge, both the chip under test and the test server calculate their keyed-hash value. A test flag is maintained storing the information whether the AES or the Hash block was used for the authentication procedure, and accordingly the other blocks are tested. This prevents redundant testing of blocks which were tested and found erroneous earlier.

Only if both symmetric-key primitives are erroneous, the testing process is aborted and a failure is reported by the server.

4.4.6 Time Overhead of the Security Mechanism

Security and efficiency of the presented test scheme come at the cost of increased test time. We hereby discuss the additional test time introduced by our secure testing methodology by splitting it into different test phases, as shown in Fig. 4.17.

- **Authentication phase.** The authentication is performed using a simple *challenge-response* protocol. The time overhead of this testing phase consists of the time needed to generate a challenge from the on-chip RNG, the time to encrypt it on both the test server and the chip and the time to transfer these values via the communication interface.
- **Testing public-key cores.** Once the SoC under test is authenticated and the test wrapper is *unlocked*, a regular scan-chain test process is used to test public key cores available in the SoC. Therefore, this testing phase does not introduce any additional time overhead compared to the regular testing process.
- **Testing symmetric key cores.** This testing phase assumes using the multiplier based BIST methodology. Since the multiplier from the public-key core is reused for this purpose to act as both test pattern generator and the signature analyzer, the time overhead consists of the following elements:
  - *Test pattern generation.* The number of clock cycles needed to generate a test pattern, N_{PAT} depends on the multiplier architecture. In the case of a fully parallel implementation, $N_{PAT} = 1$ cycle per pattern.
  - *Signature analysis.* The cycle count of this phase, N_{SIG}, depends on the multiplier architecture as well. If the multiplication is performed in a single clock cycle, $N_{SIG} = 1$.
  - *Overall.* The number of clock cycles required to test a symmetric cryptographic primitive, N_{SYM} depends on its actual hardware implementation. For instance, the number of pseudo-random patterns needed to test AES presented in [46] is 2534. Overall, the number of clock cycles required to perform the multiplier based BIST of a symmetric core is $N_{SYM} \times (N_{PAT} + N_{SIG})$.

4.4.7 Area Overhead of the Security Mechanism

Besides security, the test methodology we propose aims at achieving low area overhead. This section analyzes the cost of different components in the presented testing scenario.

Even though the modular multiplier from the existing public-key core is re-used to build the BIST infrastructure, it has to be slightly modified in order to provide the additional functionality. The overhead consists of two multiplexers placed at inputs of the multiplier (Fig. 4.18c) which allow the inputs to be supplied as operands for a regular multiplication by the CUT responses for the signature analysis and by the data for the test pattern generation. Further, a test controller that executes the test sequence from Fig. 4.17 and acts as the arbiter on the data bus (Fig. 4.7a) has to be added to the system. Finally, the comparator used in the authentication protocol and in the BIST mode to compare the golden and the calculated signature has to be added in a system. Considering the memory overhead, the presented testing scheme requires the additional non-volatile memory space to store the secret key used in the authentication protocol. Its size depends on the AES version used and typically is 128 or 256 bits.

Overall, if the total size of a cryptographic SoC is considered, it can be concluded that the area overhead introduced by the testing scheme we present is very low, ranging from 3% to 10% depending on the size of the SoC. Note that the overhead here refers only the modules that are added to provide a secure testing environment and that the standard test infrastructure, such as IEEE 1500 test wrapper which has become the industry standard for SoC testing, is not taken into analysis.

4.5 Conclusion

In this chapter, new scan attack countermeasures are proposed at the system and SoC levels. At the system level, a secure JTAG implementation based on ECC based Schnorr protocol is presented. At the SoC level, secure test wrappers are proposed combining the standard IEEE 1500 test wrapper with a lightweight block cipher or PUF-based challenge-response authentication protocol. A secure BIST strategy combining secure test wrappers and multiplier-based BIST is also proposed and evaluated as another SoC-level countermeasure. The proposed countermeasures are evaluated in terms of security, area and timing costs to draw a clear picture of the cost-benefit trade-offs involved.

Chapter 5

Conclusions and Future Work

Conclusion

In this thesis, symmetric-key and public-key cryptographic hardware implementations are evaluated in terms of their vulnerability to scan-based side-channel attacks directed at the test interface. Several commercial test compression schemes and advanced DfT structures such as X-tolerance and X-masking are studied to verify their effectiveness in protecting against scan attacks. Time compaction schemes using MISRs are evaluated in terms of their scan attack vulnerability. It is experimentally illustrated that all X-handling schemes are vulnerable to differential scan attacks, while MISR based time compaction schemes provide protection to a large extent against scan attacks provided only the final signature is observable at the outputs.

Existing side-channel countermeasures such as those that protect against power analysis attacks and fault attacks are also investigated in this thesis to evaluate whether they can be reused to protect against scan attacks. It is experimentally proven that Simple Power Analysis and Fault Attack countermeasures such as Montgomery Powering Ladder and performing dummy operations do not protect against scan attacks, whereas Differential Power Analysis countermeasures such as blinding and randomization techniques are also effective against scan attacks.

Scan attack countermeasures in the literature are evaluated to determine their effectiveness in preventing scan attacks. The cost of the countermeasures in terms of area and timing overhead is also investigated. Their effect when used together with test compression and X-handling schemes are also studied in terms of their scan attack protection. The partial scan and scan chain scrambling countermeasures are experimentally shown to be ineffective against recent differential scan attacks.

Finally, new scan attack countermeasures at different design abstraction levels are proposed. They are evaluated in terms of security, testability and test cost to understand the trade-off between these often conflicting goals inherent in all secure testing approaches for cryptographic circuits.

At the circuit or board level, a new secure JTAG implementation employing the secure Schnorr Protocol has been presented with detailed area and timing overhead costs. At the system or SoC level, secure cryptographic SoC testing approaches using lightweight block ciphers and physically unclonable functions (PUFs) are proposed for enabling test authentication mechanisms in the form of challenge-response protocols through secure test wrappers. Another novel method for testing cryptographic SoCs is presented by integrating authenticated scan based testing through Secure Test Wrappers and pseudo-random Logic BIST. The security of the presented scheme is achieved by mutually testing public and symmetric-key cryptographic modules. The symmetric-key and public-key modules are reused to provide a BIST infrastructure in order to minimize the introduced area overhead. At the gate level, a novel noise injector countermeasure to randomize the compactor outputs for test compression schemes inserted by DfT tools is proposed and evaluated in terms of security, testability and test cost.

Future Work

There are several possible avenues for future work in the area of secure testing. These avenues can be explored either as an extension of existing schemes or as a new direction of related work. We first list the extensions followed by possible new areas of related work in this domain.

In Chapter 3, we evaluate the effectiveness of other side-channel (power analysis and fault attack) countermeasures in protecting against differential scan attacks on public-key cryptographic hardware implementations in the presence of advanced DfT schemes. This approach may be extended for hardware implementations of block ciphers, such as for AES and DES. Moreover, mathematical modeling of scan attack success rates on AES can also be attempted in terms of the parameters of the scan based DfT structure such as the number of active scan chains, active slices or the distribution of the key-dependent flip-flops in the DfT infrastructure. Another possible extension is for the KATAN based secure test wrapper presented in Chapter 4. A functional test needs to be designed for testing the KATAN implementation itself. The problem of the huge size of the PUF challenge-response pair database for the PUF-based secure test wrapper also needs to be addressed in order to make its integration with the test wrapper practically feasible.

Differential scan attacks on stream ciphers [15] and Message Authentication Code (MAC) implementations employing hash functions in the presence of advanced DfT features such as test compression, X-tolerance and X-masking schemes can be a new direction for work. Security analysis of the recently proposed differential or split scan chain countermeasure [86] also needs to be performed in order to determine its effectiveness in preventing scan attacks. Hardware Trojan detection using secure DfT techniques can be investigated. Secure Memory Testing and analyzing vulnerabilities of Wireless Sensor Nodes through the test infrastructure are also possible directions of related work. Moreover, it would be worthwhile to make a practical JTAG attack on an actual set-top box decoder using the differential scan attack principles presented in this thesis. This will help in determining the vulnerability of a commercial product for the purpose of designing effective countermeasures.

Appendix A

OCSP-like public-key authentication protocol

The protocol steps are depicted in the Figure A.1.

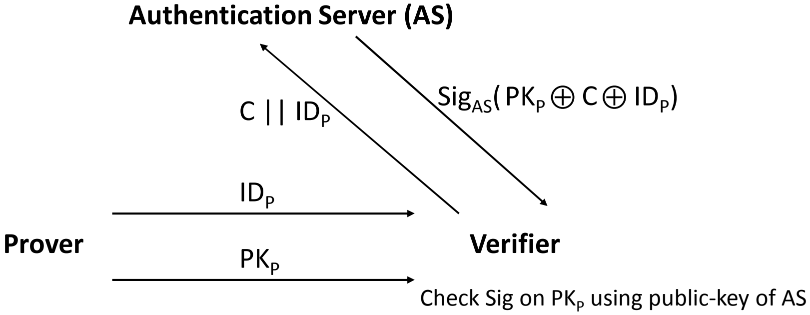


Figure A.1: Online OCSP-like public-key authentication protocol

Sig_{AS} denotes signature of the prover's public key with the private key of the Authentication Server, ID_P and PK_P are the Identity and Public-key of the Prover respectively.

The prover starts with sending its ID_P and its public key PK_P to the verifier. The verifier then initiates a call to the AS by sending a fresh random challenge C and the ID of the prover to the AS. The AS will now lookup the public key of the prover, check whether it is still valid, and if so send a signature on the XOR of PK_P, C and ID_P back to the verifier. The verifier will only accept the public key received from the prover upon reception of a valid signature by the CA on the generated challenge $PK_P \oplus C \oplus ID_P$. We are XORing the ID, challenge and the public key (instead of appending) in order to make sure that we can sign this value without first using a cryptographic hash function. In case the messages we wish to sign becomes longer than the field length of the ECC module we use, we would first have to reduce this length by employing a cryptographic hash function (and potentially cropping the result). As the implementation of such a hash function would consume too much area, we have designed our protocol to operate without a hash function.

Appendix B

ECC-based Schnorr Protocol

In the manufacturer environment scenario, A is the prover (Secure JTAG) and B is the verifier (Test server):

Pa is the public key of A and k_a is the private key of A, which are related by: $P_a = k_a$.P, where P is the initial point on the Elliptic curve (base point), which is public. k_a.P represents a point multiplication of scalar k_a with base-point P.

Goal: B wants to be ensured the identity of A, in other words A knows k_a.

Protocol:
1) A generates a random number n_a and sends an intermediate value 'T_a' (point multiplication of n_a and P) to B;
A ⇒ B: $T_a = n_a$.P
2) B generates a random number n_b and sends it to A;
A ⇐ B: n_b
3) A sends 's' to B;
A ⇒ B: s = $n_a + k_a.n_b$
Here $k_a.n_b$ represents an integer multiplication, while '+' indicates an ordinary addition.

B can verify that A is A by calculating the point multiplication of scalar s with base-point P and cross-checking it with the modular addition of 'T_a' with the point multiplication of n_b and P_a:

s.P = $T_a + n_b.P_a$
($n_a + k_a.n_b$)P = (n_a.P) + (k_a.P).n_b
n_a.P + $k_a.n_b$.P = n_a.P + $k_a.n_b$.P

Thus B verifies the identity of A by only knowing A's public key P_a.

For in-the-field updates, debug and test: A is the prover (Secure JTAG), when B is the verifier (Test server). B is the prover (Test server), when A is the verifier (Secure JTAG).

P_a is the public key of A and k_a is the private key of A, which are related by:
$P_a = k_a.\mathrm{P}$, where P is the initial point on the Elliptic curve (base point), which is public.

P_b is the public key of B and k_b is the private key of B, which are related by:
$P_b = k_b.\mathrm{P}$, where P is the initial point on the Elliptic curve (base point), which is public.

Goal: B wants to be sure that A is actually A, in other words, that A knows k_a. Similarly, A wants to be sure that B is actually B, in other words, that B knows k_b.

Protocol:
1) A generates a random number n_a, and sends it along with an intermediate value ‘T_a’ to B, which is calculated as:
A ⇒ B: $T_a = n_a.\mathrm{P}$
2) B generates two random number n_b and n_b', and sends n_b along with an intermediate value ‘T_b’ to A, which is calculated as:
A ⇐ B: $T_b = n_b'.\mathrm{P}$, n_b
3) A generates another random number n_a' and sends it along with sends ‘s’ to B, B sends ‘s_1’ to A:
A ⇒ B: $\mathrm{s} = n_a + k_a.n_b$, n_a
A ⇐ B: $s_1 = n_b' + k_b.n_a'$

B can verify that A is A by calculating:
$\mathrm{s.P} = T_a + n_b.P_a$
$(n_a + k_a.n_b).\mathrm{P} = (n_a.\mathrm{P}) + n_b.(k_a.\mathrm{P})$
$n_a.\mathrm{P} + k_a.n_b.\mathrm{P} = n_a.\mathrm{P} + n_b.k_a.\mathrm{P}$

Similarly, A can verify that B is B by calculating:
$s_1.\mathrm{P} = T_b + n_a'.P_b$
$(n_b' + k_b.n_a').\mathrm{P} = (n_b'.\mathrm{P}) + n_a'.(k_b.\mathrm{P})$
$n_b'.\mathrm{P} + k_b.n_a'.\mathrm{P} = n_b'.\mathrm{P} + n_a'.k_b.\mathrm{P}$

Thus B verifies the identity of A by only knowing A’s public key P_a, and A verifies the identity of B by only knowing B’s public key P_b.

Moreover, $n_a.n_b'.\mathrm{P}$ can be used as a session key K to encrypt all future communication between the security chip and test server. The reason behind this is that A knows $n_a.\mathrm{P}$ and n_b', while B knows n_a and $n_b'.\mathrm{P}$ from which they can construct K, but any unauthorized party cannot do so. This may be particularly useful for instance, in the case of pay-TV updates happening on the set-top box from a remote server using a network communication, where an eavesdropper can listen to the channel in between.

Bibliography

[1] *Explicit Formula Database for Jacobian Coordinates with a4=-3 for Short Weierstrass Curves.* http://www.hyperelliptic.org/EFD/g1p/auto-shortw-jacobian-3.html/.

[2] Gezel Hardware/Software codesign Environment. http://rijndael.ece.vt.edu/gezel2/.

[3] Gezel Hardware/Software Codesign Environment - Advanced Encryption Standard GEZEL Code. http://rijndael.ece.vt.edu/gezel2/examples.html#aes.

[4] JTAG: A Technical Overview. http://www.xjtag.com/support-jtag/jtag-technical-guide.php.

[5] Maestra Comprehensive Test for Satellite Testing V5. www.maestra.ca.

[6] IEEE Standard. 1149.1-1990 - IEEE Standard Test Access Port and Boundary-Scan Architecture. 1990.

[7] Guide to Understanding JTAG Fuses and Security: An Intermediate Look at the AVR JTAG Interface. AVRFreaks.net, Sept. 2002.

[8] IEEE Std 1500-2005, IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits. IEEE, 2005.

[9] Spartan-3 Generation Configuration User Guide for Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families. Xilinx UG332 (v1.6), Oct. 2005.

[10] DFT Compiler Scan User Guide, Version C-2009.06. pages 8.2–8.35. Synopsys, Sept. 2009.

[11] TetraMAX ATPG User Guide, Version C-2009.06-SP2. Synopsys, Sept. 2009.

[12] IEEE P1687 and In-Circuit Test (ICT). Asset Intertech article, June 2011.

[13] Cadence Encounter True-Time ATPG User Guide. Cadence, June 2012.

[14] S. F. Abdul-Latip, M. R. Reyhanitabar, W. Susilo, and J. Seberry. Fault Analysis of the KATAN Family of Block Ciphers. *IACR Cryptology ePrint Archive*, 2012.

[15] M. Agrawal, S. Karmakar, D. Saha, and D. Mukhopadhyay. Scan Based Side Channel Attacks on Stream Ciphers and Their Countermeasures. In *Progress in Cryptology - INDOCRYPT 2008*, pages 226–238, 2008.

[16] A. Becher, Z. Benenson, and M. Dornseif. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In *Third international conference on Security in Pervasive Computing (SPC)*, pages 104–118. Springer-Verlag, 2006.

[17] E. R. Berlekamp. *Algorithmic Coding Theory.* Mc Graw Hill, New York, 1968.

[18] I. Biehl, B. Meyer, and V. Muller. Differential Fault Analysis on Elliptic Curve Cryptosystems. pages 131–146. Advances in Cryptology - Crypto, volume 1880 of Lecture Notes in Computer Science, Springer, 2000.

[19] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology – CRYPTO 1990*, pages 2–21. Springer-Verlag, 1990.

[20] E. Biham and A. Shamir. Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer. In *Advances in Cryptology – CRYPTO 1991*, volume 576, pages 156–171. Springer-Verlag, 1991.

[21] L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[22] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *16th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 37–51. Springer-Verlag, Berlin, Heidelberg, 1997.

[23] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Kluwer Academic Publishers, 2002.

[24] R. F. Buskey and B. B. Frosik. Protected JTAG. pages 405–414. Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW), 2006.

[25] C. Cannière, O. Dunkelman, and M. Knežević. KATAN and KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Cryptographic Hardware and Embedded Systems (CHES), volume 5747 of Lecture Notes in Computer Science, Springer*, pages 272–288, Berlin, Heidelberg, 2009. Springer-Verlag.

[26] G.-M. Chiu and J.-M. Li. A Secure Test Wrapper Design Against Internal and Boundary Scan Attacks for Embedded Cores. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 20(1):126–134, Nov. 2010.

[27] M. Ciet and M. Joye. Free Randomization Techniques for Elliptic Curve Cryptography. pages 348–359. International Conference on Information, Communications and Signal Processing (ICICS), volume 2836 of Lecture Notes in Computer Science, Springer, 2003.

[28] C.J. Clark. Anti-Tamper JTAG TAP Design Enables DRM to JTAG Registers and P1687 On-Chip Instruments. pages 19–24. IEEE Symposium on Hardware-Oriented Security and Trust (HOST), 2010.

[29] H. Cohen, G. Frey, and R. Avanzi. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Chapman Hall, CRC, Boca Raton, 2006.

[30] H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 1998.

[31] D. J. Bernstein and T. Lange. Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication. Mathematics Subject Classification, 2007.

[32] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. A New Scan Attack on Elliptic Curve Cryptosystems in presence of Industrial Design for Testability Structures. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 43–48, 2012.

[33] J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. A New Scan Attack on RSA in Presence of Industrial Countermeasures. In *Third International Workshop on Constructive Side-channel Analysis and Secure Design (COSADE), volume 7275 of Lecture Notes in Computer Science, Springer*, pages 89–104, 2012.

[34] J. Da Rolt, A. Das, S. Ghosh, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. Scan Attacks on Side-Channel and Fault Attack Resistant Public-Key Implementations. *Journal Of Cryptographic Engineering (JCEN)*, 2(4):207–219, 2012.

[35] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. New Security Threats Against Chips Containing Scan Chain Structures. In *Hardware-Oriented Security and Trust (HOST)*, pages 105–110, June 2011.

[36] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. Scan Attacks and Countermeasures in Presence of Scan Response Compactors. In *IEEE European Test Symposium (ETS)*, pages 19–24, May 2011.

[37] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. Are advanced DfT structures Sufficient for Preventing Scan-Attacks? In *IEEE VLSI Test Symposium (VTS)*, pages 325–336, 2012.

[38] J. Da Rolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre. On-Chip Test Comparison for Protecting Confidential Data in Secure ICs. page 1. IEEE European Test Symposium (ETS), 2012.

[39] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[40] A. Das, J. Da Rolt, S. Ghosh, S. Seys, S. Dupuis, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. Secure JTAG Implementation using Schnorr Protocol. *Journal of Electronic Testing: Theory & Applications (JETTA)*, 29(2):193–209, 2013.

[41] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede. Security Analysis of Industrial Test Compression Schemes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 32(12), Dec. 2013.

[42] A. Das, M. Knežević, S. Seys, and I. Verbauwhede. Challenge-Response Based Secure Test Wrapper for Testing Cryptographic Circuits. In *IEEE European Test Symposium (ETS)*, May 2011.

[43] A. Das, U. Kocabas, A.-R. Sadeghi, and I. Verbauwhede. PUF-Based Secure Test Wrapper Design for Cryptographic SoC Testing. In *IEEE/ACM Design Automation and Test in Europe (DATE)*, pages 866–869, 2012.

[44] G. Di Natale, M.-L. Flottes, and B. Rouzeyre. On-Line Self-Test of AES Hardware Implementations. WDSN, 2007.

[45] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.*, 38(1):97–139, Mar. 2008.

[46] M. Doulcier, M. L. Flottes, and B. Rouzeyre. AES-Based BIST: Self-Test, Test Pattern Generation and Signature Analysis. In *4th IEEE International Symposium on Electronic Design, Test and Applications*. IEEE, 2008.

[47] D.-Z. Du, B. Lu, H. Ngo, and P. M. Panos. Steiner Tree Problems. *Encyclopedia of Optimization*, pages 2451–2464, 2001.

[48] B. Ege, A. Das, L. Batina, and I. Verbauwhede. Security of Countermeasures Against State-of-the-Art Differential Scan Attacks. In *Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE), Co-located with IEEE European Test Symposium (ETS)*, 2013.

[49] B. Ege, A. Das, S. Ghosh, and I. Verbauwhede. Differential Scan Attack on AES with X-Tolerant and X-Masked Test Response Compactor. In *IEEE Euromicro Conference on Digital System Design (DSD)*, pages 545–552, 2012.

[50] O. Erdinç, H. G., and B. Sunar. Towards Robust Low Cost Authentication for Pervasive Devices. In *International Conference on Pervasive Computing and Communications (PERCOM)*, pages 170–178. IEEECSP, Mar. 2008.

[51] J. Fan, B. Gierlichs, and F. Vercauteren. To Infinity and Beyond: Combined Attack on ECC Using Points of Low Order. In *Cryptographic Hardware and Embedded Systems (CHES), volume 6917 of Lecture Notes in Computer Science*, pages 143–159. Springer, 2011.

[52] J. Fan and I. Verbauwhede. An updated survey on secure ecc implementations: Attacks, countermeasures and cost. In *Cryptography and Security*, volume 6805 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2012.

[53] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems (CHES), volume 2162 of Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[54] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled Physical Random Functions. In *Computer Security Applications Conference (ACSAC)*, pages 149–160. IEEE, 2002.

[55] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury. Petrel: Power and Timing Attack Resistant Elliptic Curve Scalar Multiplier Based on Programmable Arithmetic Unit. *IEEE Transactions on Circuits and Systems - I*, 58(11):1798–1812, 2011.

[56] B. Gierlichs. *Statistical and Information-Theoretic Methods for Power Analysis on Embedded Cryptography.* PhD thesis, Katholieke Universiteit Leuven – Faculty of Engineering, Jan 2011. I. Verbauwhede and B. Preneel (supervisors).

[57] D. Golic. New Methods for Digital Generation and Postprocessing of Random Data. *IEEE Transactions on Computers*, 55(10):1217–1229, Oct. 2006.

[58] L. Greenemeier. iPhone Hacks Annoy AT&T but are Unlikely to Bruise Apple. Scientific American, http://www.scientificamerican.com/article.cfm?id=iphone-hacks-annoy-at, 2007.

[59] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[60] C. Hartung, J. Balasalle, and R. Han. Node Compromise in Sensor Networks: The Need for Secure Systems. Technical Report CU-CS-990-05, Dept of Computer Science, Univ. of Colorado at Boulder, 2005.

[61] D. Hely, F. Bancel, M.-L. Flottes, and B. Rouzeyre. Test Control for Secure Scan Designs. In *IEEE European Test Symposium (ETS)*, pages 190–195, 2005.

[62] D. Hely, F. Bancel, and B. Rouzeyre. Secure Scan Techniques: A Comparison. In *IEEE International Symposium on Online Testing Symposium (IOLTS)*, pages 119–124, 2006.

[63] D. Hely, M.-L. Flottes, F. Bancel, B. Rouzeyre, N. Berard, and M. Renovell. Scan design and Secure Chip [Secure IC Testing]. In *IEEE International Online Test Symposium (IOLTS)*, pages 219–224, 2004.

[64] M. Hutter, M. Feldhofer, and T. Plos. An ECDSA processor for RFID Authentication. pages 189–202. Workshop on RFID Security (RFIDSec), volume 6370 of Lecture Notes in Computer Science, Springer, 2010.

[65] M. Inoue, T. Yoneda, M. Hasegawa, and H. Fujiwara. Balanced Secure Scan: Partial Scan Approach for Secret Information Protection. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 27(2):99–108, April 2011.

[66] T. Itoh and S. Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases. *Information and Computation*, 78:171–177, 1988.

[67] J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. pages 292–302. Workshop on Cryptographic Hardware and Embedded Systems (CHES), volume 1717 of Lecture Notes in Computer Science, Springer, 1999.

[68] M. Joye and J. Devigne. Binary Huff Curves. pages 340–355. RSA Conference Cryptographers' Track (CT-RSA), volume 6558 of Lecture Notes in Computer Science, Springer, 2011.

[69] M. Joye and C. Tymen. Protections against Differential Analysis for Elliptic Curve Cryptography. pages 377–390. Cryptographic Hardware and Embedded Systems (CHES), volume 2162 of Lecture Notes in Computer Science, Springer, 2001.

[70] M. Joye and S.-M. Yen. The Montgomery Powering Ladder. pages 291–302. *Workshop on Cryptographic Hardware and Embedded Systems (CHES), volume 2523 of Lecture Notes in Computer Science, Springer*, 2003.

[71] D. Karaklajic, A. Das, and I. Verbauwhede. Secure Mutual Testing Strategy for Cryptographic SoCs. *Elsevier VLSI Integration Journal, to be submitted*, 2013.

[72] D. Karaklajić, M. Knežević, and I. Verbauwhede. Multiplier based Built-in Self-Test for Cryptographic Applications. Technical report, KU Leuven, 2010.

[73] T. Kern and M. Feldhofer. Low-Resource ECDSA Implementation for Passive RFID Tags. pages 1236–1239. ICECS, 2010.

[74] N. Koblitz. Elliptic Curve Cryptosystem. *Math. Comp.*, 48:203–209, 1987.

[75] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology – Crypto, volume 1109 of Lecture Notes in Computer Science, Springer*, pages 104—-113. Springer-Verlag, 1996.

[76] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology – Crypto, volume 1666 of Lecture Notes in Computer Science, Springer*, pages 388–397. Springer-Verlag, 1999.

[77] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic. Securing Designs against Scan-Based Side-Channel Attacks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(4):325–336, Oct.-Dec. 2007.

[78] Y.-K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic Curve-Based Security Processor for RFID. *IEEE Transaction on Computers*, 57(11):1514–1627, 2008.

[79] D. Lim, J. W. Lee, B. Gassend, E. G. Suh, M. van Dijk, and S. Devadas. Extracting Secret Keys from Integrated Circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(10):1200–1205, Oct. 2005.

[80] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Extracting Secret Keys from Integrated Circuits. *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, 13:1200–1205, 2005.

[81] C. Liu and Y. Huang. Effects of Embedded Decompression and Compaction Architectures on Side-Channel Attack Resistance. In *IEEE VLSI Test Symposium (VTS)*, pages 461–468, 2007.

[82] Y. Liu, K. Wu, and R. Karri. Scan-Based Attacks on Linear Feedback Shift Register Based Stream Ciphers. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 16(2O):20:1–20:15, March 2011.

[83] R. Maes. *Physically Unclonable Functions: Constructions, Properties and Applications.* PhD thesis, Katholieke Universiteit Leuven – Faculty of Engineering, Aug 2012. I. Verbauwhede (supervisor).

[84] R. Maes and I. Verbauwhede. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer-Verlag, 2010.

[85] B. C. Mames, M. Ciet, and M. Joye. Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IACR Cryptology ePrint Archive*, 2003.

[86] S. Manich, M. S. Wamser, and G. Sigl. Improving Scan Path Test Security Using Differential Chains. In *First Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*, 2013.

[87] E. J. Marinissen, S. K. Goel, and M. Lousberg. Wrapper Design for Embedded Core Test. pages 911–920. IEEE International Test Conference (ITC), 2000.

[88] E. J. Marinissen and Y. Zorian. IEEE Std 1500 Enables Modular SoC Testing. *IEEE Design & Test of Computers*, pages 8–17, Jan./Feb. 2009.

[89] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[90] Mentor Graphics. Silicon Test and Yield Analysis Whitepaper - High Quality Test Solutions for Secure Applications. April 2010.

[91] V. Miller. Uses of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1985.

[92] P. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44:519–521, 1985.

[93] P. Montgomery. Speeding the Pollard and Elliptic Curve Methods for Factorizations. *Mathematics of Computation*, 48:243–264, 1987.

[94] R. Nara, K. Satoh, M. Yanagisawa, and T. Ohtsuki. Scan-Based Attack against Elliptic Curve Cryptosystems. In *Asia and South Pacific Design Automatic Conference (ASP-DAC)*, pages 407–412, 2010.

[95] R. Nara, K. Satoh, M. Yanagisawa, and T. Ohtsuki. State-Dependent Changeable Scan Architecture against Scan-Based Side Channel Attacks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1867–1870, 2010.

[96] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa. Scan-Based Side-Channel Attack Against RSA Cryptosystems Using Scan Signatures. *IEICE Transaction on Fundamentals of Electronics Communications and Computer Sciences*, 93(12):2481–2489, 2010.

[97] National Bureau of Standards. Data Encryption Standard. *Federal Information Processing Standards Publication 46*, January 1977.

[98] National Bureau of Standards. Digital Signature Standard (DSS). *Federal Information Processing Standards Publication 186*, May 1994.

[99] National Bureau of Standards. Secure Hash Standard. *Federal Information*, Processing Standards Publication 180-1, April 1995.

[100] National Bureau of Standards. Secure Hash Standard. *Federal Information*, Processing Standards Publication 180-3, October 2008.

[101] National Institute of Standards and Technology. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. *Federal Information*, Special Publication 800-67, May 2008.

[102] S. Neophytou, M. K. Michael, and S. Tragoudas. Efficient Deterministic Test Generation for BIST Schemes with LFSR Reseeding. pages 43–50. IEEE International On-Line Testing Symposium (IOLTS), 2006.

[103] F. Novak and A. Biasizzo. Security Extension for IEEE Std. 1149.1. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 22(3):301–303, June 2006.

[104] K. Park, S.-G. Yoo, and J. Kim. JTAG Security System Based on Credentials. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 26(5):549–557, 2010.

[105] L. Pierce and S. Tragoudas. Multi-Level Secure JTAG Architecture. pages 208–209. IEEE International Symposium on Online Testing Symposium (IOLTS), 2011.

[106] J. M. Pollard. A Monte Carlo Method for Factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975.

[107] J.-J. Quisquater and D. Samyde. Electromagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *Smart Card Programming and Security (E-smart), volume 2140 of Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[108] J. Rajski and J. Tyszer. Accumulator-Based Compaction of Test Responses. *IEEE Transactions on Computers*, 42(6):643–650, June 1993.

[109] J. Rajski and J. Tyszer. Multiplicative Window Generators of Pseudorandom Test Vectors. In *European Design and Test Conference*, pages 42–48, 1996.

[110] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded Deterministic Test. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 23(5):776–792, 2004.

[111] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, and G. Mrugalski. Embedded Deterministic Test for Low Cost Manufacturing Test. In *IEEE International Test Conference (ITC)*, pages 301–310, 2002.

[112] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian. Embedded Deterministic Test for Low cost Manufacturing Test. In *IEEE International Test Conference (ITC)*, pages 301–310, 2002.

[113] E. Rippel. Security Challenges in Embedded Designs. Discretix Technologies Ltd., Design & Reuse article, http://www.design-reuse.com/articles/20671/security-embedded-design.html.

[114] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[115] K. Rosenfeld and R. Karri. Attacks and defenses for jtag. *IEEE Design & Test of Computers*, 27(1):36–47, 2010.

[116] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *ACM Conference on Computer and Communications Security (ACM CCS)*, pages 237–249, New York, NY, USA, 2010. ACM.

[117] A. Satoh and T. Inoue. ASIC-Hardware-Focused Comparison for Hash Functions MD5, RIPEMD-160, and SHS. pages 532–537. International Conference on Information Technology: Coding and Computing (ITCC), 2005.

[118] C. Schnorr. Efficient Identification and Signatures for Smart Cards. pages 239–252. Advances in Cryptology - Crypto, volume 435 of Lecture Notes in Computer Science, Springer, 1990.

[119] A. Schubert and W. Anheier. On Random Pattern Testability of Cryptographic VLSI Cores. *Journal of Electronic testing: Theory and applications (JETTA)*, 16:185–192, 2000.

[120] G. Sengar, D. Mukhopadhyay, and D. Chowdhury. Secured Flipped Scan-Chain Model for Crypto-Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 26(11):2080–2084, Nov. 2007.

[121] Y. Shi, N. Togawa, M. Yanagisawa, and T. Ohtsuki. Design for Secure Test - A Case Study on the Pipelined Advanced Encryption Standard. pages 149–152. IEEE International Symposium on Circuits and Systems (ISCAS), 2007.

[122] S. Skorobogatov and R. Anderson. Optical Fault Induction Attacks. pages 2–12. Workshop on Cryptographic Hardware and Embedded Systems (CHES), volume 2523 of Lecture Notes in Computer Science, Springer, 2002.

[123] L. Song and L. Hu. Improved Algebraic and Differential Fault Attacks on the KATAN Block Cipher. In *Information Security Practice & Experience Conference (ISPEC)*, pages 372–386. volume 7863 of Lecture Notes in Computer Science, Springer, 2013.

[124] F. P. Stroele. Bit Serial Pattern Generation and Response Compaction using Arithmetic Functions. In *16th IEEE VLSI Test Symposium (VTS)*, pages 78–84, 1998.

[125] E. G. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 9–14. IEEE, June 2007.

[126] F. Veljkovic, V. Rozic, and I. Verbauwhede. Low-Cost Implementations of On-the-Fly Tests for Random Number Generators. In *IEEE/ACM Design Automation and Test in Europe (DATE)*, pages 959–964, 2012.

[127] B. Škorić, P. Tuyls, and W. Ophey. Robust Key Extraction from Physical Uncloneable Functions Applied Cryptography and Network Security. In *Applied Cryptography and Network Security (ACNS) 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 99–135, Berlin, Heidelberg, 2005. Springer Berlin / Heidelberg.

[128] L.-T. Wang, C.-W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon).* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, ISBN: 0123705975, 2006.

[129] Z. Wang, K. Chakrabarty, and S. Wang. Integrated LFSR Reseeding, Test Access Optimization, and Test Scheduling for Core-Based System-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(8):1251–1264, Aug. 2009.

[130] P. Wohl, J. Waicukauski, R. Kapur, S. Ramnath, E. Gizdarski, T. Williams, and P. Jaini. Minimizing the Impact of Scan Compression. In *IEEE VLSI Test Symposium (VTS)*, pages 67–74, 2007.

[131] P. Wohl, J. A. Waicukauski, S. Patel, and M. B. Amin. X-Tolerant Compression and Application of Scan-ATPG Patterns in a BIST Architecture. In *IEEE International Test Conference (ITC)*, pages 727–736, 2003.

[132] B. Yang, K. Wu, and R. Karri. Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard. In *IEEE International Test Conference (ITC)*, pages 339–344, 2004.

[133] B. Yang, K. Wu, and R. Karri. Secure Scan: A Design-for-Test Architecture for Crypto Chips. In *IEEE/ACM Design Automation Conference (DAC)*, pages 135–140, 2005.

[134] B. Yang, K. Wu, and R. Karri. Secure Scan: A Design-for-Test Architecture for Crypto Chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 25(10):2287–2293, Oct. 2006.

[135] S. Yen and M. Joye. Checking before Output Not be Enough Against Fault-Based Cryptanalysis. *IEEE Transactions on Computers*, 49(9):967–970, 2000.

[136] M. Yilmaz and K. Chakrabarty. Seed Selection in LFSR-Reseeding-Based Test Compression for the Detection of Small-Delay Defects. pages 1488–1493. IEEE/ACM Design Automation and Test in Europe (DATE), 2009.

Curriculum Vitae

Amitabh Das was born in Ranchi, India on August 15, 1981. He received the B.Tech degree in Electronics and Communication Engineering from Kalyani Government Engineering College, University of Kalyani, West Bengal, India in 2003 and M.Tech degree in Instrumentation and Electronics Engineering from Jadavpur University, Kolkata, India in 2009. From 2003 to 2007, he worked as an Assistant Systems Engineer at Tata Consultancy Services (TCS), Kolkata, India, and as an Electronics and Instrumentation Engineer at Bharat Heavy Electricals Limited (BHEL), India. In July 2009, he joined the COSIC research group in the Electrical Engineering Department at the Katholieke Universiteit Leuven (KU Leuven), Belgium as a provisional Doctoral student and started his PhD program in the embedded security sub-group with emphasis on "Differential Scan-Based Side-Channel Attacks and Countermeasures". His research interests include secure design-for-testability, electronic design automation, hardware cryptography, embedded systems, and hardware/software co-design and co-simulation.

List of Publications

International Journals

[1] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, “Security Analysis of Industrial Test Compression Schemes,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 32, no. 12, 2013.

[2] A. Das, J. Da Rolt, S. Ghosh, S. Seys, S. Dupuis, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “Secure JTAG Implementation using Schnorr Protocol,” Journal of Electronic Testing: Theory & Applications (JETTA), Springer Science and Business Media New York 2013, Inc., vol. 29, no. 2, pp. 193-209, 2013.

[3] J. Da Rolt, A. Das, S. Ghosh, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “Scan Attacks on Side-Channel and Fault Attack Resistant Public-Key Implementations,” Journal Of Cryptographic Engineering (JCEN), Springer-Verlag Berlin Heidelberg, vol. 2, no. 4, pp. 207-219, 2012.

[4] L. Uhsadel, M. Ullrich, A. Das, D. Karaklajic, J. Balasch, I. Verbauwhede, and W. Dehaene, “Teaching HW/SW co-design with a public key cryptography application”, IEEE Transactions on Education (TE), 2013.

International Conferences and Workshops

[1] L. Batina, A. Das, B. Ege, E B. Kavun, N. Mentens, C. Paar, I. Verbauwhede, and T. Yalcin, “Dietary Recommendations for Lightweight Block Ciphers: Power, Energy and Area Analysis of Recently Developed Architectures”, Workshop on RFID Security (RFIDSec 2013), July 2013.

[2] S. Ghosh, A. Kumar, A. Das, and I. Verbauwhede, “On the Implementation of Unified Arithmetic on Binary Huff Curves”, Workshop on Cryptographic Hardware and Embedded Systems (CHES 2013), pp. 349-364, August 2013.

[3] J. Da Rolt, A. Das, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "A New Scan Attack on Elliptic Curve Cryptosystems in presence of Industrial Design for Testability Structures," IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT 2012), pp. 43-48, October 2012.

[4] B. Ege, A. Das, S. Ghosh, and I. Verbauwhede, "Differential Scan Attack on AES with X-Tolerant and X-Masked Test Response Compactor," IEEE Euromicro Conference on Digital System Design (DSD 2012), pp. 545-552, September 2012.

[5] J. Da Rolt, A. Das, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "A New Scan Attack on RSA in Presence of Industrial Countermeasures," Third International Workshop on Constructive Side-channel Analysis and Secure Design (COSADE 2012), Springer LNCS 7275, pp. 89-104, May 2012.

[6] A. Das, U. Kocabas, A-R. Sadeghi, and I. Verbauwhede, "PUF-based Secure Test Wrapper Design for Cryptographic SoC Testing," IEEE/ACM Design Automation and Test in Europe (DATE 2012), pp. 866-869, March 2012.

[7] A. Das, M. Knezevic, S. Seys, and I. Verbauwhede, "Challenge-Response based Secure Test Wrapper for testing Cryptographic Circuits," IEEE European Test Symposium (ETS 2011), May 2011.

[8] B. Ege, A. Das, L. Batina, and I. Verbauwhede, "Security of Countermeasures Against State-of-the-Art Differential Scan Attacks," Workshop on Trustworthy Manufacturing and Utilization of Secure Devices (TRUDEVICE 2013), co-located with IEEE European Test Symposium (ETS 2013), 6 pages, May 2013.

[9] A. Das, M. Knezevic, S. Seys, and I. Verbauwhede, "Challenge-Response based Secure Test Wrapper for testing Cryptographic Circuits," Workshop on Information and System Security (WISSec 2010), 11 pages, November 2010.

Journals under review

[1] J. Da Rolt, A. Das, G. Di Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test versus Security: Past, Present and Future," IEEE Transactions on Emerging Topics in Computing (TETC) Special issue on Nanoscale secure, reliable architectures, submitted, under review, 12 pages, 2013.

Technical Reports

[1] D. Karaklajic, A. Das, and I. Verbauwhede, "Secure Mutual Testing Strategy for Cryptographic SoCs," COSIC internal report, 9 pages, 2013.

[2] A. Das, B. Ege, and I. Verbauwhede, “Security Analysis of Industrial Test Compression Schemes,” COSIC internal report, 7 pages, 2012.

[3] J. Da Rolt, A. Das, G. D. Natale, M-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “Security Analysis Tool for Scan-Based Side-Channel Attacks on Cryptographic Circuits,” COSIC internal report, 10 pages, 2011.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING (ESAT)
COMPUTER SECURITY AND INDUSTRIAL CRYPTOGRAPHY (COSIC)
Kasteelpark Arenberg 10, box 2452
B-3001 Heverlee
Amitabh.Das@esat.kuleuven.be
http://www.esat.kuleuven.be/cosic