

March 2020

EFFICIENT HARDWARE PRIMITIVES FOR SECURING LIGHTWEIGHT SYSTEMS

Siva Nishok Dhanuskodi

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2



Part of the [VLSI and Circuits, Embedded and Hardware Systems Commons](#)

Recommended Citation

Dhanuskodi, Siva Nishok, "EFFICIENT HARDWARE PRIMITIVES FOR SECURING LIGHTWEIGHT SYSTEMS" (2020). *Doctoral Dissertations*. 1821.
<https://doi.org/10.7275/re76-xh53> https://scholarworks.umass.edu/dissertations_2/1821

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

EFFICIENT HARDWARE PRIMITIVES FOR SECURING LIGHTWEIGHT SYSTEMS

A Dissertation Presented

by

SIVA NISHOK DHANUSKODI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2020

Electrical and Computer Engineering

© Copyright by Siva Nishok Dhanuskodi 2020

All Rights Reserved

EFFICIENT HARDWARE PRIMITIVES FOR SECURING LIGHTWEIGHT SYSTEMS

A Dissertation Presented

by

SIVA NISHOK DHANUSKODI

Approved as to style and content by:

Daniel E. Holcomb, Chair

Wayne P. Burleson, Member

Russell G. Tessier, Member

Charles Weems, Member

Christopher V. Hollot, Department Head
Electrical and Computer Engineering

DEDICATION

To my parents Dhanuskodi S. and Parvathi B.

ACKNOWLEDGMENTS

I would like to first of all thank my advisor Professor Daniel Holcomb. His technical astuteness, patient guidance and unwavering support throughout my PhD have gone a long way in making this thesis happen. He has been always been willing to give a lot of his time and energy. The many interactions I have had with him have shaped my research goals as well as honed my technical skills. This thesis would not have been possible without his constant friendly support and encouragement.

I would like to thank the members of my thesis committee for their valuable feedback and time. A special thanks to Professor Burleson, Professor Kundu and Professor Koren for their guidance, discussions and critique in group meetings. Their personal interest and experience have helped me improve research quality. I would like to acknowledge the faculty at ECE department in general for an inspiring and conducive learning atmosphere.

It was great to work with several people all these years: Xiang Li, Shahrzad Keshavarz, Harshavardhan Ramanna, Vinay Patil, Arunkumar Vijayakumar and Raghavan Kumar. I really enjoyed our collaborations, technical discussions and thank you for your support at several stages of my graduate life. A special mention to some of my friends who always stood by me with encouragement and support: Arunachalam Annamalai, Sankara Narayanan Rajapandian, Meenakshi Sundaram Bhaskaran, Surendran Subramanian and Badri Krishna Kumar.

The list would not be complete without thanking my family for their selfless love, support and care. Their entire life has been an offering to my brother and I, more than I could ever ask for. Yoga has been a huge support in every aspect of my life, I would like to express my heartfelt gratitude to Sadhguru.

ABSTRACT

EFFICIENT HARDWARE PRIMITIVES FOR SECURING LIGHTWEIGHT SYSTEMS

FEBRUARY 2020

SIVA NISHOK DHANUSKODI

B.E., ANNA UNIVERSITY - MADRAS INSTITUTE OF TECHNOLOGY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Daniel E. Holcomb

In the era of IoT and ubiquitous computing, the collection and communication of sensitive data is increasingly being handled by lightweight Integrated Circuits. Efficient hardware implementations of cryptographic primitives for resource constrained applications have become critical, especially block ciphers which perform fundamental operations such as encryption, decryption, and even hashing. We study the efficiency of block ciphers under different implementation styles. For low latency applications that use unrolled block cipher implementations, we design a glitch filter to reduce energy consumption. For lightweight applications, we design a novel architecture for the widely used AES cipher. The design eliminates inefficiencies in data movement and clock activity, thereby significantly improving energy efficiency over state-of-the-art architectures. Apart from efficiency, vulnerability to implementation attacks are a concern, which we mitigate by our randomization capable lightweight AES architecture. We fabricate our designs in a commercial 16nm FinFET technology and

present measured testchip data on energy consumption and side channel resistance. Finally, we address the problem of supply chain security by using image processing techniques to extract fingerprints from surface texture of plastic IC packages for IC authentication and counterfeit prevention. Collectively these works present efficient and cost effective solutions to secure lightweight systems.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	v
ABSTRACT	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiv
CHAPTER	
INTRODUCTION	1
1. BACKGROUND	5
1.1 Block ciphers	5
1.1.1 AES	5
1.1.2 SIMON	6
1.2 Energy efficiency in block ciphers implementations	7
1.3 Side Channel Attacks	9
1.3.1 Differential Power Analysis and metrics	9
1.3.2 Existing countermeasures	11
1.3.3 Logic State Based Leakage Power Analysis	12
1.3.4 Remote Side Channel Attack on FPGAs	14
1.4 Supply Chain Security	19
2. EFFICIENCY IN UNROLLED BLOCK CIPHERS	21
2.1 Glitches and glitch filtering	21
2.2 Checkpointing to improve energy efficiency	23
2.3 Evaluation of Checkpointing	26
2.3.1 Methodology	26

2.3.2	Comparison of Average Switching Rates	27
2.3.3	Energy Comparison in Fully Unrolled Designs	29
2.3.3.1	SIMON-128	29
2.3.3.2	AES-128	31
2.3.4	Optimal Placement of Checkpoints for Glitch Filtering	32
2.3.5	Checkpointing in Partially Unrolled Designs	33
2.3.5.1	SIMON-128	34
2.3.5.2	AES-128	35
2.3.6	Area Cost of Checkpointing	36
2.3.7	Power Gating	38
2.3.8	Voltage Scaling	39
2.4	Summary	40
3.	EFFICIENCY IN LIGHTWEIGHT AES	42
3.1	Novel architecture	44
3.1.1	Improved clocking	44
3.1.2	Register renaming	45
3.1.3	Implementation	49
3.2	Microarchitectural Randomization	55
3.2.1	Enable Generator design for Word and Byte shuffling	55
3.2.2	Mix Columns design to handle permutation	57
3.2.3	Sequencing of Rounds and Key Expansion	59
3.3	Evaluation	61
3.3.1	Efficiency and overheads	61
3.3.2	Susceptibility to Side Channel Attacks	64
3.4	Summary	66
4.	TESTCHIP	68
4.1	Design methodology	68
4.1.1	RTL design and synthesis	68
4.1.2	Physical design	70
4.1.3	Chip packaging and Printed Circuit Board design	74

4.2	Chip Testing	75
4.2.1	Efficiency, power	76
4.2.2	Side Channel resilience	79
4.3	Summary	83
5.	PACKAGE IDENTIFICATION	85
5.1	Transfer Molding for IC packaging	87
5.2	COUNTERFOIL anti-counterfeiting scheme	90
5.2.1	Enrollment	91
5.2.2	Verification	91
5.2.3	Attacker Capabilities and Security Considerations	92
5.3	Image Processing and Analysis	95
5.3.1	Aruco marker labels and detection of ROI	95
5.3.2	Feature Enrollment	96
5.3.3	Feature Verification	98
5.3.3.1	Feature matching and RANSAC based homography computation	98
5.3.3.2	Projection and Scoring	99
5.4	Evaluation	100
5.4.1	Package Authentication	102
5.4.2	Runtime	105
5.4.3	Practicality and Costs	105
5.4.4	Algorithm Difference	107
5.4.5	Camera Differences	107
5.4.6	Varying Magnification and Lighting	108
5.5	Further Investigation of Fingerprints	109
5.5.1	Testing Resilience of Fingerprints	109
5.5.2	Testing Fingerprint Uniqueness	110
5.5.2.1	Scoring under Controlled Alignment	111
5.5.2.2	PUF-like evaluation using Pixel Intensity	112
5.5.2.3	PUF-like evaluation using Feature Distance	114
5.5.3	Additional Package Types	115
5.6	Summary	117

6. CONCLUSION	118
LIST OF PUBLICATIONS.....	119
BIBLIOGRAPHY	120

LIST OF TABLES

Table	Page
1.1 Simulated Dynamic and static power consumption of a WDDL-AND gate.	13
2.1 Breakdown of E_{enc} (pJ/bit) in fully unrolled SIMON-128. Glitch filters are added after every round in Round Gating and our checkpointing work	31
2.2 Breakdown of E_{enc} (pJ/bit) in fully unrolled AES-128. Glitch filters are added after every round in Round Gating and our checkpointing work	32
2.3 E_{enc} (pJ/bit) comparison in SIMON-128 between optimal checkpointing and the baseline design for various degrees of unrolling.	36
2.4 E_{enc} (pJ/bit) comparison in AES-128 between optimal checkpointing and the baseline design for various degrees of unrolling.	37
2.5 Area penalty of proposed glitch filtering scheme in units of gate equivalents. Even in absolute terms, the area cost of checkpointing is significantly higher in SIMON-128 than in AES-128 because the larger number of rounds requires a larger number of checkpoints, even though the checkpoints are only applied at every second round.	37
3.1 Physical registers to be enabled in each clock cycle	49
3.2 Table illustrates the operation of the pipelined MixColumns (see Fig. 3.10) for two different orderings of the input bytes. At the end of the four cycles, the same values exist in the registers for both orderings, but their locations differ. The permutation step associates the appropriate register value to each output signal.	59
3.3 Comparison of energy efficiency of four AES designs all implemented by us in the same 16nm technology.	62

3.4	Comparison of area of four AES designs all implemented by us in the same 16nm technology.	63
3.5	Comparison of performance. Throughput obtained at 300MHz clock.	63
4.1	Performance comparison of AES designs with testchip measurements obtained at 20MHz clock.	79
4.2	Measurements to Disclosure using Hamming Distance DPA on the 8-bit AES designs	83
5.1	Quantitative comparison of different feature-detecting methods. Plot at right shows the ROC plot from which the area-under-curve is computed. All four algorithms are configured to use 1,000 keypoints per mm^2 for this comparison.	107

LIST OF FIGURES

Figure		Page
1.1	Structure of AES round function.	6
1.2	Structure of AES 128-bit Key Expansion [87].	7
1.3	SIMON round function and key schedule [13].	8
1.4	Illustration of a DPA attack on AES.	11
1.5	Schematics for LLPA.	14
1.6	A successful attack using LLPA.	15
1.7	FDPA Attack setup	16
1.8	FDPA in action.	19
2.1	Schematic of latch-based checkpoints for glitch filtering.	25
2.2	Timing diagram of glitch filter operation, annotated with the number of switching events happening at each point in the circuit for SIMON-128.	25
2.3	Increasing pulse width of enable signal to tolerate variations.	25
2.4	SIMON-128 energy per encryption histogram for 100 random inputs.	27
2.5	Comparison of the average toggle rate of the output signals of each round of SIMON-128 for four different implementation styles.	29
2.6	Contribution of each round to the overall energy per encrypted bit in four different implementation styles of fully unrolled SIMON-128.	30
2.7	Energy/encryption breakdown in fully unrolled implementations using checkpointing after every round.	32

2.8	Energy efficiency varies with the spacing between checkpoints in fully unrolled designs. Performing more computation between checkpoints reduces checkpointing energy, but allows more data switching to occur	34
2.9	Energy breakdown of E_{enc} for each round in fully unrolled SIMON-128 in the optimal configuration of checkpointing every second round.	35
2.10	Power gating to reduce leakage power consumption.	39
2.11	Energy per encryption at different supply voltages in SIMON. Dotted lines represent leakage energy.	40
3.1	Proposed 8-bit architecture	45
3.2	Timing of control signals for a quarter of one round. enB signals enable bytes to be read from physical registers into the datapath S-Box, and enW signals allow round outputs to be written back to physical registers on falling edge.	46
3.3	Illustration of register renaming	48
3.4	Enable Generator	52
3.5	Schematic of Round Function	54
3.6	Schematic of Key Expansion	54
3.7	Randomization-capable 8-bit AES architecture. Additional details of redesigned Enable Generator and MixColumns circuits are shown in Figs. 3.9 and 3.10.	55
3.8	Shuffled orders in which bytes can be processed in our architecture. Depending on the value of the word offset and byte offset, each of the 16 state bytes could be processed in any of the 16 cycles of the round.	57
3.9	Enable Generator for Randomized architecture (compare Fig. 3.4). Solid, dotted and red lines indicate 4-bit, 1-bit and 2-bit signals respectively.	58
3.10	MixColumns for Randomized architecture.	60

3.11	Plots showing DPA attack on 4 key bytes for the 8-bit designs. Top plot shows differential power traces for the 8-bit renaming design and the bottom plot shows the same for the 8-bit randomized design. Green line corresponds to correct key guess and red lines correspond to incorrect key guesses.	65
3.12	CDF of MTD for the different designs.	65
4.1	Block level view of designs implemented on testchip.	69
4.2	Different stages of physical design of 8-bit Randomized AES.	71
4.3	Layout showing all four AES designs.....	73
4.4	Examples of sign-off DRC violations on finished design that were manually fixed.	73
4.5	Images of full chip with other designs from colleagues on same die. All designs have isolated power domains.....	74
4.6	Chip packaged in a Flip Chip Ball Grid Array	76
4.7	AES chip test setup.	77
4.8	Power and energy comparison of AES designs based on testchip measurements.	78
4.9	Effect of voltage scaling on efficiency.	80
4.10	Differential and DPA traces with Hamming distance leakage model.	81
4.11	Differential Power Analysis	82
5.1	Transfer molding is the mechanism used for packaging most high-volume microchips.	88
5.2	Protocol for package fingerprinting. Trusted enroller labels each package and then enrolls it by extracting and then signing a set of keypoints associated with the package. Verifier compares the enrolled keypoints against the package to determine whether the package is consistent with its label.	89

5.3	Size of features extracted from images of package surfaces using OpenCV implementation of ORB algorithm as discussed in Sec. 5.3.2	94
5.4	Image of chip with affixed marker. The position of enrollment ROI is shown by the blue box, and the callout shows the keypoints extracted from the ROI. The ROI that would be used for verification is the smaller red box. The size and position of both ROIs are defined relative to the marker, as shown by annotations in yellow.	96
5.5	Pixel distance between the expected location of a keypoint (according to homography) and the location of its nearest neighbor in feature space. The spike at left shows points for which the nearest neighbor is found in the expected location. The points that are sufficiently close to be counted as inliers are the ones colored red.	100
5.6	Three examples of matching between enrollment keypoints (square in upper left) and verification image of the same chip package instance, where the verification image differs in zoom and orientation. White square on chip package is the identified region of interest for verification. Each line corresponds to a keypoint match from enrollment to verification (Sec. 5.3.3.2).	101
5.7	Experimental setup. Left side of workbench used for enrollment, right side used for verification. Separate camera are used for enrollment and verification. Middle of image shows the population of chips with labels affixed.	102
5.8	CDF of number of inliers using each model of camera.	103
5.9	Receiver Operating Characteristic curves show ability to distinguish enrolled chips from other chips created from a different mold than the enrolled chip, or from the same mold that produced the enrolled chip.	104
5.10	Runtime of verification procedure, broken down by processing task, for different sizes of ROI. Keypoint density is held constant at $1,000/mm^2$. The increase in keypoints for the larger ROI results in a higher runtime, but also increases the number of matching points that are found. Runtime can be traded against accuracy by adjusting the ROI size.	106

5.11	Histograms showing increase in number of inliers in AS6C6264 SRAM when same ViTiny cameras are used for both enrollment and verification.	108
5.12	Histogram of inliers in AS6C6264 SRAM under two alternative lighting intensities (nominal is 800 lux) and one alternative zoom.	109
5.13	Reduction in inliers for chip AS6C6264 after spending time in rock tumbler. Images of chip are included to give a sense of the amount of wear caused.....	111
5.14	Inlier CDFs for SRAMs under controlled alignment.	112
5.15	PUF-like evaluation on raw pixel intensity data.	113
5.16	Average distance in feature space for same-position keypoint pairs.	115
5.17	Evaluation of package surface fingerprints across a range of package types.	116

INTRODUCTION

Advances in the chip industry have enabled, at the time of this thesis, production of Integrated Circuits (ICs) with transistor channel lengths as small as a few nanometers. Miniaturization has opened avenues for ultra-lightweight systems such as the Internet-of-Things (IoT). Some of the multi-billion dollar [115] applications of IoT include smart homes and cities, operations and equipment optimization in factories, wearable/implantable medical devices and autonomous vehicles.

Ubiquitous computing often involves sharing personal and confidential data over the Internet. As a result security of IoT devices becomes important especially in critical applications like healthcare. Some of the unique challenges in IoT security include limited hardware resources, stringent power budgets, and cost constraints. Lightweight IoT devices like Radio Frequency Identification (RFID) tags and wireless sensor nodes are typically battery powered with energy budgets of less than a microjoule per bit of data processed [115]. Physically accessibility of these devices to an attacker opens a variety of security threats, both passive and active attacks.

The basic security services required of a cryptosystem are confidentiality, integrity, message authentication and nonrepudiation [87]. Block ciphers form an important building block in offering these services. Symmetric key encryption algorithms encrypt and authenticate data using a secret key. Advanced Encryption Standard (AES) is the most widely used symmetric key based block cipher. Dedicated hardware implementations of AES are used in millions of hardware chips worldwide to encrypt large blocks of data with better performance and power than can be achieved in software implementations.

Implementing cryptographic primitives such as AES for IoT systems is challenging due to cost constraints and power budgets. Designers exploit the symmetry in the operations of the AES algorithm to reduce the number of hardware units implemented and reuse them over time to save area. However block ciphers have data dependencies which these make lightweight implementations inefficient as a significant amount of energy is spent in moving data around to work around the dependencies.

Apart from efficiency concerns, AES can be vulnerable to *Side Channel Attacks* (SCAs) that target weaknesses of the hardware implementation to extract the secret key. Passive SCAs exploit correlation between computed data and the power consumption of hardware implementing AES to retrieve the secret key. Active attacks inject a fault in the AES computation and retrieve the secret key by comparing the outputs of faulty and faulty-free computations. Lightweight devices for IoT are especially susceptible to SCAs as they have low background noise power, are physically accessible to the attacker, and have low budget for defenses.

Besides the implementation weaknesses, the supply and distribution channels of ICs present a large, diverse and vulnerable attack surface. Counterfeit ICs such as phony parts, recycled, and remarked chips have made their way into critical defense and avionics systems. With ubiquitous computing, the problem of counterfeit parts becomes increasingly critical. IoT systems are often combining chips from different sources, and a single bad chip can compromise the entire system.

In this dissertation, we present a background to further understand some of the aforementioned problems, then provide novel and efficient solutions. The dissertation ranges on topics from silicon to package, covering the entire stack of a hardware system. We study the reasons for energy inefficiencies in state-of-the-art block ciphers and design micro-architectures that greatly mitigate inefficiencies under different scenarios. We support our claims with data from a testchip in an advanced commercial 16nm FinFET technology. We also present a novel IC authentication technique that

uses computer vision techniques to prevent counterfeits and can be deployed atop current infrastructure at almost no cost.

My specific contributions include:

- Developed glitch filtering techniques that allow partially and fully unrolled block ciphers to have an energy efficiency that is competitive with serialized implementations
- Designed a novel microarchitecture for lightweight AES implementations that minimizes data movement and clock activity to improve energy efficiency
- Enabled randomization of sub-round operations in lightweight AES architecture to mitigate side channel susceptibility
- Taped out AES designs in a commercial 16nm FinFET technology chip and tested efficiency and side channel resilience of the designs
- For the first time, showed that individual chip packages can be recognized and authenticated using intrinsic surface features extracted using low cost cameras and image processing

The remainder of the dissertation is structured as follows. Chapter 1 provides the necessary background about block ciphers, side channel attacks and supply chain security. Efficiency of unrolled block ciphers is addressed in chapter 2 through a new glitch filter design (published in *Trans. on Computers* 2017 [30]) enabling unrolled ciphers to be competitive with serialized designs but with the drawback of significant area costs. We next explore a novel lightweight AES architecture with side channel resilience in chapter 3. Our design significantly outperforms state-of-the-art and we published our work in ISLPED 2017 [32] and ISVLSI 2019 [31]. We further taped out our designs in 16nm FinFET technology and chapter 4 describes the design and evaluation of our testchip containing four AES designs. In chapter 5 we discuss our

novel IC package authentication methodology to verify provenance of ICs. We present concluding remarks in chapter 6.

CHAPTER 1

BACKGROUND

1.1 Block ciphers

Block ciphers are cryptographic primitives that encrypt and decrypt data, typically within a larger encryption mode of operation to help achieve security goals of confidentiality and authenticity for a cryptosystem. Block ciphers can also be used as part of hash functions. Typically, a block cipher algorithm iterates over a round function for a specified number of times using different round keys generated by the key schedule function. Substitution and permutation are two common operations found in the round functions of block ciphers. Symmetric key block ciphers such as DES and AES are used for data encryption due to their simple design and performance. Asymmetric algorithms such as RSA and ECC employ complex mathematical operations and are therefore used for key exchange and digital signatures. Now we describe the block ciphers AES and SIMON, as we will be using them in this thesis.

1.1.1 AES

The Advanced Encryption Standard (AES) is a ubiquitous encryption standard [91] based on the Rijndael cipher. It was standardized by NIST in 2001 to replace DES following an open competition. AES uses a number of iterated rounds, 10 in the case of a 128-bit key strength, to transform a block of plaintext into a corresponding block of ciphertext. Each round (Fig. 1.1) operates on 128 bits of state, and uses a 128-bit round key to generate the next state from the current state. The major components of the round function are SubBytes, ShiftRows, MixColumns, and addition of the round key. The SubBytes function uses an S-Box circuit to apply the

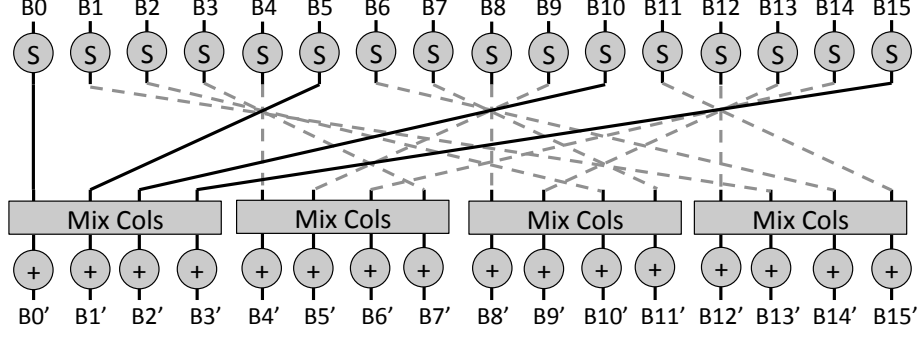


Figure 1.1: Structure of AES round function.

same byte-wise substitution function to each of the 16 state bytes. ShiftRows operation reorders the bytes. MixColumns operates on a 4-byte input $\{s_3, s_2, s_1, s_0\}$ and produces a 4-byte output $\{m_3, m_2, m_1, m_0\}$ as per Eq. 1.1. Finally the round key is added (XORed) to the output of MixColumns to create the next state that will be used as the input to the next round. The 128-bit round keys are expanded from a single 128-bit key input as shown in Fig. 1.2. Round key 0 is the input key and $RC[i] = x^{i-1} \bmod (x^8 + x^4 + x^3 + x + 1)$ is the Round Constant of round i .

$$\begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (1.1)$$

1.1.2 SIMON

SIMON is a lightweight Feistel cipher suitable for resource constrained systems, and we use the SIMON-128 variant [13]. SIMON-128 uses a 128-bit key, 128-bit data, and requires 68 rounds for each encryption. A Feistel structure is symmetric in nature and offers the advantage of similarities in the encryption and decryption operations, which in turn reduces the required resources for a hardware implementation. The round function (Fig. 1.3a) is constructed to be extremely small in hardware and easy

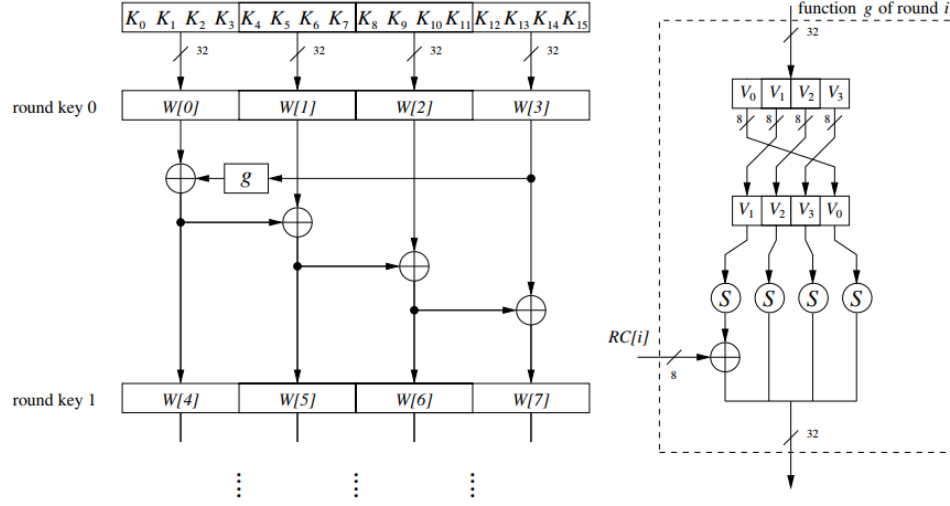


Figure 1.2: Structure of AES 128-bit Key Expansion [87].

to serialize without sacrificing software performance [13]. It consists of only the following operations: bitwise XOR, bitwise AND and left circular shift S^j by j bits. The round key k_i is generated from the input key according to the key schedule operation shown in Fig. 1.3b. The operations in the key schedule are similar to the ones in the round function. To eliminate vulnerabilities to cryptanalysis attacks such as slide attack [18] that exploits cyclic nature of key schedule and rotational attacks that exploit correlations between bit-rotated pairs of inputs [62], a 1-bit round constant is employed in each round of the key schedule from the following 62-period sequence $z_2 = 62'b \ 101011111011100000011010010011000101000010001111110010110110011$.

1.2 Energy efficiency in block ciphers implementations

Block ciphers are almost always implemented as components of a larger overall system-on-chip design, and this prevents the block cipher from being freely optimized independently of the other SoC components. For example, the block cipher will have to use the same fabrication process and supply voltage as the other components, and typically will share a common clock frequency to avoid clock generation and

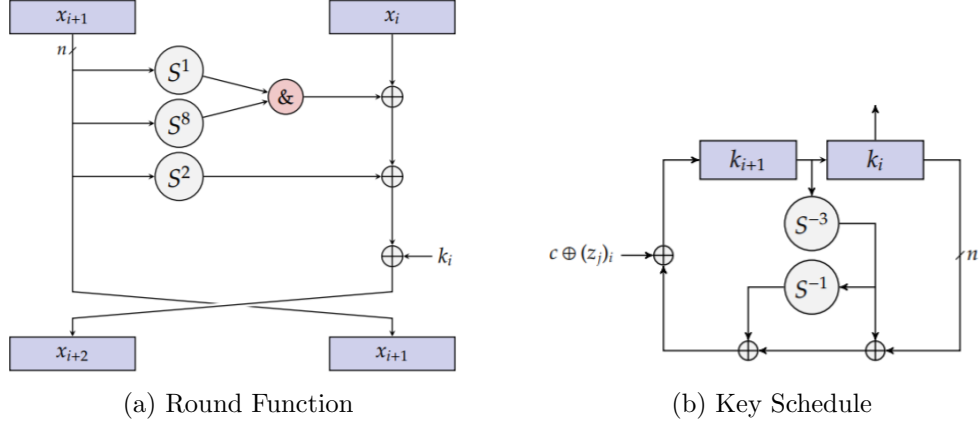


Figure 1.3: SIMON round function and key schedule [13].

clock domain crossing. Therefore, any attempt at optimizing block ciphers may be constrained by these chip-scale implementation decisions.

Depending on the chip-scale requirements and constraints, the block cipher rounds can be implemented through sequential reuse of a single combinational block for each round, or they can be unrolled. If a design is serialized (no unrolling), one round function is computed in each clock cycle, and the number of cycles needed to encrypt a block is the same as the number of rounds in the block cipher algorithm. Yet at slow clock frequencies, the clock period may far exceed the critical path delay of a block cipher round. The latency of the block-cipher is then being increased unnecessarily due to the serialization of the round function. Unrolling has been explored in literature as a technique to instantiate multiple round functions per clock cycle and eliminate energy spent in loop control elements such as registers and multiplexers. However the energy savings are minimal and are offset by the increase in glitching power that comes with unrolling; we further explore this in Ch. 2.

For resource constrained applications like IoT, lightweight implementations of block ciphers have been explored. Symmetry in sub-round operations has been leveraged by temporal reuse of limited hardware units to save area in ciphers such as AES [78, 47]. For example, if a single AES S-box (Fig. 1.1) is implemented in hardware

it can be reused to operate on different bytes across clock cycles. In this case one round is completed in 16 cycles. The increase in latency due to narrow datapath operation is acceptable in these non-performance critical applications. However, significant energy inefficiencies exist due to the storage of intermediate results and data movement which are explored and addressed in Ch. 3. Though lightweight ciphers such as SIMON have been proposed they are not used as extensively as AES.

1.3 Side Channel Attacks

Although encryption algorithms and protocols have been developed to provide security, hardware implementations can leak valuable information to an attacker. For example, data dependence in power consumption of a device can be exploited as a side-channel to extract its secret key. Such a security attack is termed a Side Channel Attack (SCA). Some examples of passive side-channels include encryption time, power consumption or electromagnetic radiation emanating from device that are correlated to the computed data [104]. Active attacks on the other hand inject faults into the computation by varying supply voltage, clock frequency, or exposing the device to lasers, and subsequently analyze faulty and fault-free outputs to retrieve the secret key [57, 104]. Combined active and passive attacks also exist [95].

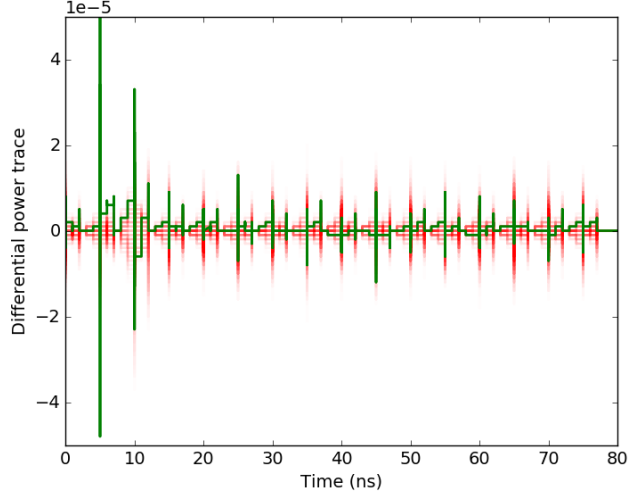
1.3.1 Differential Power Analysis and metrics

Power side-channel attacks such as Simple Power Analysis (SPA) [64], Differential Power Analysis (DPA) [64] and Correlation Power Analysis (CPA) [22] leverage data dependency in power consumption to extract secret information. In a DPA attack, one would capture power traces T_0, T_1, \dots, T_{m-1} while the encryption algorithm is running and record the corresponding ciphertexts C_0, C_1, \dots, C_{m-1} . An internal node value b in the computation that is dependent on a few bits of round key and ciphertext is then chosen as a selection function. In AES, the attack is performed on values that

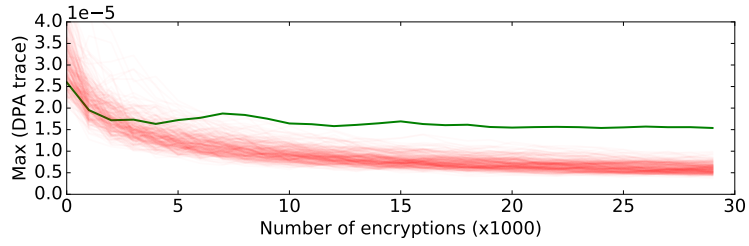
depend on only a single byte of round key at a time. Now the attacker guesses the key bits, and for each guess K_s he partitions the power traces T_0, T_1, \dots, T_{m-1} based on the computed value of the selection function $D(C_i, b, K_s)$. The attacker then computes a differential trace Δ_D as shown in Eq. 1.2. For the correct key guess, with enough power measurements a peak in the differential trace is observed at the time when the selection function is computed in hardware, due to the correlation between the predicted selection function and the differential average of power traces (Δ_D). For incorrect key guesses, a random partitioning of power traces results in a differential average that approaches zero with enough measurements.

$$\Delta_D[j] = \frac{\sum_{i=1}^m D(C_i, b, K_s) T_i[j]}{\sum_{i=1}^m D(C_i, b, K_s)} - \frac{\sum_{i=1}^m (1 - D(C_i, b, K_s)) T_i[j]}{\sum_{i=1}^m (1 - D(C_i, b, K_s))} \quad (1.2)$$

For example in a DPA attack on AES, by making power measurements the secret key of AES in a smart card is broken by using S-box input value as selection function. The attack works despite not being able to observe the S-box inputs. In Fig. 1.4a the differential power trace values for different key guesses are plotted. The correct key guess (green line) has a higher DPA peak than incorrect key guesses (red lines). With enough measurements only the correct key guess has a consistently high DPA peak as seen in Fig. 1.4b. **Measurements to Disclosure (MTD)** [107] is the number of measurements required to distinguish the correct key guess from incorrect ones. MTD is defined as the cross-over point between the differential peak of the correct key guess and the maximum differential peak of all the wrong key guesses. In Fig. 1.4b we see an MTD of about 13K encryptions. Even though we demonstrate DPA with AES, side-channel attacks can also break insecure implementations of other cryptographic algorithms such as RSA and DES [63].



(a) DPA power trace



(b) MTD

Figure 1.4: Illustration of a DPA attack on AES.

1.3.2 Existing countermeasures

A number of circuit-level countermeasures exist for side-channels. Circuit-level countermeasures are imperfect but can drastically decrease the signal-to-noise ratio of the information leaked through the side channel. One family of countermeasures tries to modify the power delivery system so that the power consumed by the encryption circuit will not be externally visible to the attacker. Switched-capacitor designs can isolate the sensitive computation from main power by sourcing its current from a capacitor and then discharging the capacitor to a fixed value before replenishing it [108]. On-chip low-dropout regulators [100] can also be used to obscure the power consumption of the sensitive circuit.

A second family of circuit-level countermeasures are modified logic styles that try to achieve a power consumption that is independent of the values being computed. SABL and WDDL [105] are differential logic styles that always compute both true and complement values of each node so that power consumption is unrelated to the computed value; these logic styles require careful routing to balance the loading on the differential signals, but given enough measurements can still be attacked due to unavoidable small imbalances in the differential routing [107]. Masked dual-rail precharge logics seek to avoid routing constraints by using a random mask to reduce correlation between power consumption and processed data [90, 89], but are costly in area.

Another approach to preventing side channel attacks, which can be complementary to circuit-level countermeasures, is hiding the timing of the computation from the attacker. An attacker that does not know when a certain key byte is processed will have difficulty aligning the power traces in the way that is required for DPA attacks. Timing randomization can easily be accomplished by inserting idle delays, but idle delays can be identified in the power trace and removed by the attacker. Randomly inserting dummy encryptions or dummy rounds between meaningful computations adds a delay that cannot easily be detected and removed in post-processing, but consumes significant power and latency and requires added complexity around the cipher. We further explore efficient timing randomization in Sec. 3.2.

1.3.3 Logic State Based Leakage Power Analysis

Side-channel attacks have primarily focused on dynamic power as it has typically been assumed to be the main contributor of the power consumption. However with today’s technology scaling, leakage power has become a significant contributor as well. While the countermeasures such as WDDL, SABL mitigate the information leakage

Table 1.1: Simulated Dynamic and static power consumption of a WDDL-AND gate.

Input State	Absolute currents		Normalized to state 00	
	Dynamic (μA)	Static (μA)	Dynamic	Static
00	9.83	0.174	1.00	1.00
01	9.80	0.231	1.00	1.33
10	9.73	0.206	0.99	1.18
11	9.82	0.238	1.00	1.37

through dynamic power consumption, the secret data can still be vulnerable through the analysis of leakage power.

Leakage Power Analysis (LPA) attacks are relatively new and have not been as thoroughly explored as dynamic power attacks. The overall leakage power of a cryptographic core is related to the secret data [43] and DPA techniques can be applied to leakage power in the presence of process variations [70]. Effectiveness of a Hamming Weight based LPA attack on cryptographic cores has been studied for various side-channel resistant logic styles in the presence of noise and process variations [6].

WDDL and other side-channel resistant logic styles offer resistance against dynamic power attacks. However, WDDL still has a significant data dependence in the static (leakage) power consumption (Tab. 1.1). Dynamic power can be made arbitrarily balanced (using SABL for instance) but static power cannot. We leverage the data dependence of static power to predict the total leakage power consumed by a circuit and from this extract secret information such as cryptographic keys. We name our scheme Logic state based Leakage Power Analysis (LLPA) [33].

We assume the attacker has knowledge of the circuit’s implementation (in terms of logic gates) and is able to observe primary outputs and measure power consumption. Note that our assumption that an attacker knows the exact gates implemented inside of a design is increasingly a very practical assumption, as this technique has been employed in security research [86, 101], and one can even purchase this information commercially through companies such as Chipworks. Even if a design uses camou-

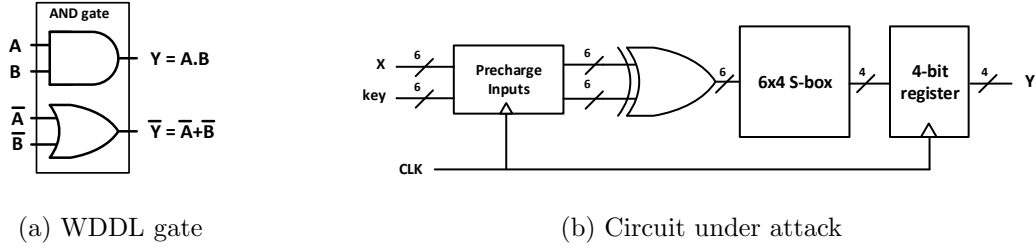


Figure 1.5: Schematics for LLPA.

flagged gates [92], the functionality of gates can often be extracted [77, 71]. Also, note that the internal state is unobservable in the attack model and is predicted with design knowledge and known data inputs.

To demonstrate our methodology, we choose to attack in simulation a 6x4 S-box used in DES and retrieve the secret key. The schematic of our circuit is shown in Fig. 1.5b. The static current of each gate type for every gate input combination is obtained in a pre-characterization step. For each data input the attacker guesses the key value K_j and computes the logical input state of each gate in the design. Under a key guess, state dependent static currents of all gates are summed up to compute a predicted static power P_j for the entire design. This is repeated for all key guesses $j = 0, 1, \dots, 63$ and the correlation C_j between static power predictions P_j and measurement M is computed per Eq. 1.3. With enough measurements only the correct key guess should have a consistently high correlation value. Fig. 1.6 shows a successful attack with an MTD of 1071 measurements.

$$C_j = \rho_{M, P_j} = \frac{\text{covariance}(M, P_j)}{\sigma_M \sigma_{P_j}} \quad (1.3)$$

1.3.4 Remote Side Channel Attack on FPGAs

As part of my thesis, I collaborated with students from the Reconfigurable Computing Group at UMass to investigate a remote side channel attack on FPGAs which I describe in this section. The FPGA implementations described in this section were

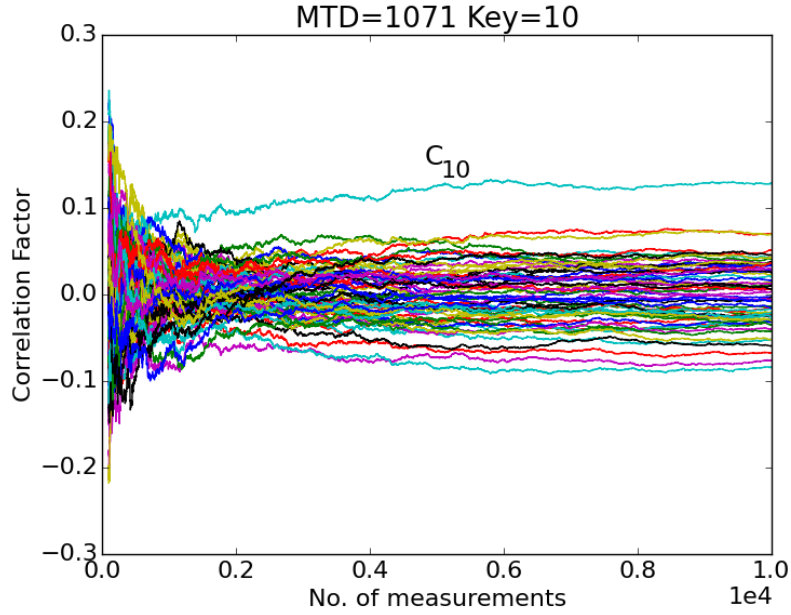


Figure 1.6: A successful attack using LLPA.

done by my colleagues. I was responsible for developing the attack methodology and related software code. This work was published in 2018 [93].

Field Programmable Gate Arrays (FPGAs) are quickly growing in importance in a variety of computing spaces including cloud computing and embedded platforms (automotive, military, and aerospace). As FPGAs grow in size and complexity, it is apparent that numerous applications from independent users may simultaneously reside in a single FPGA device. This use of *multi-tenant* FPGAs opens the door to numerous potential attack vectors on unsuspecting co-located FPGA circuits. Although FPGA devices in cloud computing environments such as Microsoft Catapult [4] and Amazon EC2 F1 [1] are currently dedicated to a specific application, the growing capabilities of FPGAs makes it easy to envision single-FPGA platforms containing multiple independent applications created by completely separate entities.

The discovery of a covert communication channel between neighboring FPGA long wires (also called "long lines") has the potential to dramatically change the threat level of multi-tenant FPGAs. In a comprehensive set of experiments, Giechaskiel *et*

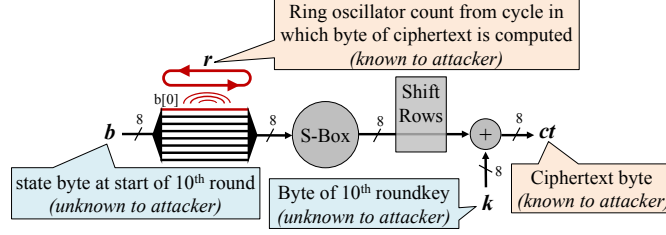


Figure 1.7: FDPA Attack setup

al. [42] showed that the logic value carried on a long wire influences the delay of both its immediate neighbor and a long wire in the same channel two wires away. When a logic 1 value is carried on a wire (the transmitter), the delay in the neighboring wire (the receiver) is reduced relative to when a logic 0 is transmitted. We leverage this information leakage by implementing a Ring Oscillator (RO) near a long wire (the victim), and use a binary counter to measure the RO's frequency: we increment a 32-bit count value at every positive edge of RO clock for a fixed time duration. The count values will depend on the data being transmitted on the victim wire.

Our Frequency DPA (FDPA) attack draws inspiration from DPA. As an example, we demonstrate the working of FDPA on AES to retrieve the secret key. The relevant portion of the AES final round circuit for attacking a key byte, using information leaked from a single wire, is shown in Fig. 1.7. Recall that the final round of the AES algorithm performs bitwise substitution (S-Box), shift rows, and key addition using XOR, but it omits the mix columns operation. The output of the final round is the ciphertext, which is public information. To set up the attack scenario for recovering a key byte, the attacker chooses as the victim any bit of S-Box input that is routed on a long wire; in Fig. 1.7, bit 0 of the S-Box input is chosen as the victim. The ring oscillator is then routed next to this signal so that its oscillation count in each clock cycle will depend slightly on the value of the S-Box input bit.

Using the ring oscillator as a sensor, the attacker monitors many encryptions to collect information for the side channel attack. For each of n encryptions performed,

the attacker records the ciphertext byte and the ring oscillator count during the cycle the ciphertext byte was produced; we denote these two quantities as ct_i and r_i respectively for the i^{th} encryption. After n encryptions, the attacker has a collection of measured oscillator count and ciphertext pairings $(r_0, ct_0), (r_1, ct_1), \dots, (r_{n-1}, ct_{n-1})$. Among the 256 possible key byte values, the attacker correctly identifies the key byte used in the circuit based on side channel measurements as follows. For each key guess k_j (i.e. $k_0 \dots k_{255}$), the attacker computes an S-Box input value $b_{i,j}$ for each of the $i \in [0, n - 1]$ measurements using Eq. 1.4 to invert the circuit's round key addition and S-Box computation.

$$b_{i,j} = S^{-1}(ct_i \oplus k_j) \quad (1.4)$$

By inverting the S-Box function under key guess k_j , the attacker now knows what S-Box input value would have induced ciphertext ct_i if the key byte was in fact k_j . For key guess k_j , the computed values at the S-Box input in the n encryptions would be denoted $b_{0,j}, b_{1,j}, \dots, b_{n-1,j}$. The predicted S-Box inputs each contain a specific prediction on the value of the victim wire (bit 0 of the S-Box input), and we check for its effect on the oscillator counts to know whether k_j is the correct key byte value. The attacker next partitions the n measurements into two subsets according to whether the victim wire would have a 0 or 1 value under the key guess k_j – one subset contains all the measured RO counts (r_i) for encryptions when the victim would have a 1 value, and the other subset contains all the measured RO counts when the victim would have a 0 value. The attacker then uses the average RO counts of the two subsets to confirm or refute his guess that k_j is the key byte value as follows:

- If the key byte is in reality k_j , then partitioning according to key guess k_j is accurately partitioning the data based on whether the victim is 0 or 1. The average RO count will tend to be higher in the subset of encryptions that predict a 1-value for the victim wire, and lower in the subset of encryptions that predict

a 0-value. Observing a sufficient difference between the average RO counts in the two subsets confirms that the partition is meaningful, and thus supports the hypothesis that the correct key byte value is k_j .

- If the key byte is not in reality k_j , then partitioning according to key guess k_j is arbitrary and not correlated to the computation of the circuit. Because the partition is arbitrary, each subset will contain a similar proportion of RO counts taken when the victim wire is 0 and 1. In this case, the average RO count from each subset will be similar, and the difference between the average RO counts of the two sets will approach 0 with enough data. Observing no difference between the average RO counts of the two subsets therefore serves to refute the hypothesis that the key byte value is k_j .

Fig. 1.8a shows graphically how a collection of RO counts can confirm or refute a key guess. The attacker in this case collects 500 RO counts and corresponding ciphertexts; the RO counts for the measurements are shown in the top plot of Fig. 1.8a. The middle plot shows which of the counts are predicted, according to the correct key guess, to occur when the victim wire is 1 and 0. We can see that, in measurements when the key guess predicts the victim wire to have a 1 value, the RO counts tend to be higher. The significant difference in average RO counts gives an attacker confidence that the key guess is correct. The lower plot of Fig. 1.8a uses an incorrect key guess to predict the 1 and 0 values of the victim wire. Using this key guess there is no difference between the average RO counts, indicating to an attacker that the key guess is not the correct one. Using this approach, with enough side channel data, the attacker will be able to identify the correct key byte guesses, even when the difference between the average RO counts is quite small. Successful attacks were performed on different FPGAs at clock frequencies upto 10MHz and the MTDs are shown in Fig. 1.8b.

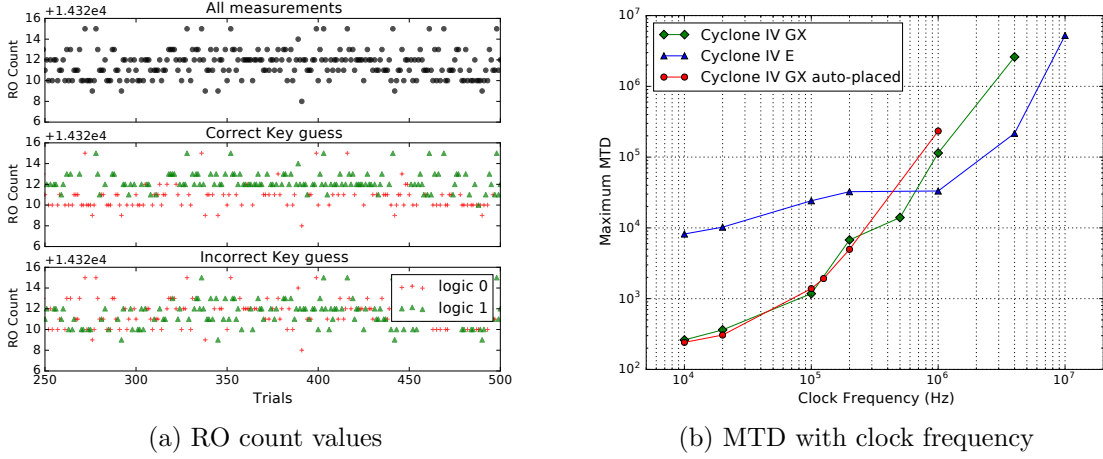


Figure 1.8: FDPA in action.

1.4 Supply Chain Security

ICs take on critical roles in today’s society, but the supply and distribution channels for ICs are vulnerable to a variety of security threats. One such threat is counterfeit parts, which are a significant and increasing threat to the reliability of electronic systems. Counterfeits are defined by the US Department of Defense as “unauthorized copies and previously used parts that are made to look new, and are sold as new” [94]. Misrepresented ICs such as speed binned parts that are remarked to a higher speed grade to increase selling price [88] can also be considered counterfeits. Prior research claims that recycled and remarked chips together make up 80% of all counterfeiting incidents [45]. These types of counterfeit parts are enabled by a lack of traceability through distribution channels as parts change hands through resellers and system integrators. DARPA notes that chain-of-custody solutions are unworkable for securing distribution due to components that may change hands 15 times before final installation [67]. Our work addresses this critical security problem by giving an approach for securing parts through distribution without chain-of-custody.

Estimates variously place the direct losses from electronics counterfeiting at \$3B-\$7.5B [55], and the potential risk due to counterfeiting at \$100B-\$200B [88, 84]. The

most commonly counterfeited electronics are said to be analog ICs, microprocessors, memories, programmable logic, and discrete transistors [52, 45]. Documented cases of counterfeit parts include purported microcontrollers that were found to be remarked voltage regulators [103], four instances of counterfeit parts in the Avionics Systems of C-27J aircraft [98], and refurbished flash memory devices in Terminal High-Altitude Area Defense (THAAD) mission computers that led to a recall of 50 systems [94]. Counterfeit parts such as these present clear security risks which we address in Ch. 5.

CHAPTER 2

EFFICIENCY IN UNROLLED BLOCK CIPHERS

Unrolling a block cipher is the process of instantiating multiple rounds of the algorithm combinationally to be completed within each clock cycle. Unrolling allows the result to be computed in fewer cycles at the cost of increased area of the combinational circuit. Unrolling also saves some amount of register energy, as energy is not spent storing signals at the output of each round like the fully serialized case. The unrolling of block ciphers as an energy optimization technique has been explored in a number of recent works [58, 11]. Switching power, especially due to glitches, is a challenge in unrolled implementations of a block cipher. In this chapter we present an efficient latch-based glitch filter for unrolled designs that reduces energy per encryption by an order of magnitude over a straightforward unrolled implementation, and by 28-45% over the best existing glitch filtering schemes. We explore the optimal number of glitch filters to use for minimizing total energy, and provide estimates of the area cost. Power gating to reduce leakage power and reuse of computed key enable unrolled designs to be more efficient than serialized ones. We demonstrate our approach on the SIMON-128 and AES-128 block ciphers.

2.1 Glitches and glitch filtering

The limiting factor in energy minimization of block ciphers is switching energy. This is especially true in unrolled block ciphers because combinational logic glitches at the input of each round diffuse through the round to cause more glitches at the output of the round. Leakage power is small relative to switching power for typical

clock periods and technologies used in low power designs [58]. Fundamentally, glitches occur because of mismatched arrival times of gate inputs. This causes the gate output to switch once when the first input arrives, and then switch again when the next input arrives. These two switching events then propagate to many other nodes and cause more switching events in a cascading fashion.

Several techniques to filter glitches have been proposed in literature. Pipelining [19, 114] stops glitches because they cannot propagate through a register, as a register can change its output value only once per clock cycle upon arrival of the clock transition. Gate-freezing [14] stalls the computation in a gate by using an NMOS footer transistor to filter 1-to-0 transitions. The stalled gate is allowed to compute only when its inputs have reached their final state. The scheme has a limitation in that it allows 0-to-1 transitions to pass through a stalled gate. Retiming [81] by moving or adding flip-flops in the datapath to high activity nodes that have a large fanout can reduce glitches and save power. Yet another approach is delay balancing to equalize input arrival times at a gate and reduce the number of output switching events [66, 51].

An AND gate based glitch filtering scheme (Round Gating) has recently been proposed in [9]. The output signals of each round in this scheme are gated by AND gates that wait on an enable signal. The enable signal is derived from a delayed clock such that it goes high to propagate the round outputs through the AND gates only after they have stopped glitching and become stable. A drawback of this scheme is that the enable signals must be reset low between the end of one computation and the start of the next in order to stop propagation of the glitches in the next operation. When the enable signals go low, waves of 0s propagate forward from the glitch filters and through the circuit to charge and discharge the nodes in the round functions similar to a normal computation of the round function. Effectively, resetting the glitch filters is thus causing a second, unnecessary, power-wasting computation to

occur. State-retaining barriers [83] provide a mechanism for preventing this power-wasting computation. The use of state-retaining elements is effective for reducing glitching in FPGAs [69, 27, 35].

2.2 Checkpointing to improve energy efficiency

Combinational checkpointing is a microarchitectural technique to increase energy efficiency in a combinational circuit by filtering glitches. We propose a new standard-cell compatible glitch filtering mechanism [29, 30] as shown in Fig. 2.1. The topology is similar to that of round gating using AND gates [9], except that the glitch filtering element consists of a positive latch implemented using a multiplexer (MUX) at the output of the round function. The purpose of the filter is to make sure that any glitching activity from its input is not propagated to its output.

The operation of the filter is as follows. The MUX holds on to its previous output value when the enable (select) signal is low, and becomes transparent when enable is high. This causes the latch to be transparent only during the enable pulse. The enable pulse is generated at the rising edge of the clock as the AND of the clock signal and a delayed inverted version of clock. The enable pulse is propagated to the glitch filters combinatorially with timing controlled by adding a delay element per round function. If the propagation delay of the delay element (t_d) is greater than the critical delay of a round function (t_r), then round output r_i stabilizes before the rising edge of signal en_i , so the latches only become transparent after the glitching has stopped. Therefore, when this timing condition ($t_d > t_r$) is satisfied, glitches generated in round i do not propagate through the glitch filters to round $i + 1$. Because the latch stays open for the duration of the enable pulse, the circuit will function correctly as long as the round outputs stabilize before the falling edge of en_i , but the circuit will not filter any glitches that arrive when the latch is open, and the glitch filter will not have the intended effect.

The timing waveform for a single round of SIMON-128 is shown in Fig. 2.2. When the enable signal pulses at the first glitch filter, the stable outputs of round $i - 1$ propagate through round i and cause a total of 122 transitions on the 128 round output signals. The round outputs wait for the enable signal to arrive at the second glitch filter, and upon its arrival, only 60 transitions occur on the inputs of round $i + 1$; these 60 transitions are single transitions on 60 of the 128 signals, which is close to the expected number of bits that would differ between two uncorrelated 128-bit signals. In this case, the filter has prevented all the spurious glitches from propagating across rounds.

The propagation delay of the round function (t_r) can be determined through static timing analysis, and propagation delay of the delay element (t_d) can be configured to exceed t_r by a conservative 20% margin. This timing margin provides resilience against PVT variations and ensures that the enable signal always arrives after the round computation is complete. If the enable pulse arrives at a checkpoint before the round computation has completed, glitches will propagate through the open latch, but the computation can still be functionally correct as long as the round outputs stabilize before the falling edge of the enable pulse closes the latch. Correct functionality requires $t_r < t_d + w$, where w is the enable pulse width, and glitch free operation requires $t_r < t_d$. Therefore, a more aggressive t_d can be chosen by widening the enable pulse to tolerate variations as illustrated in Fig. 2.3. Widening the enable pulse allows performance improvements by paying a small (glitching) energy cost, and can be useful if only a few instances of the round function are slow due to process variations. The enable pulses can be made very wide as the input to a round function does not change until the next clock cycle, and the enable pulses can even be delayed versions of the clock signal.

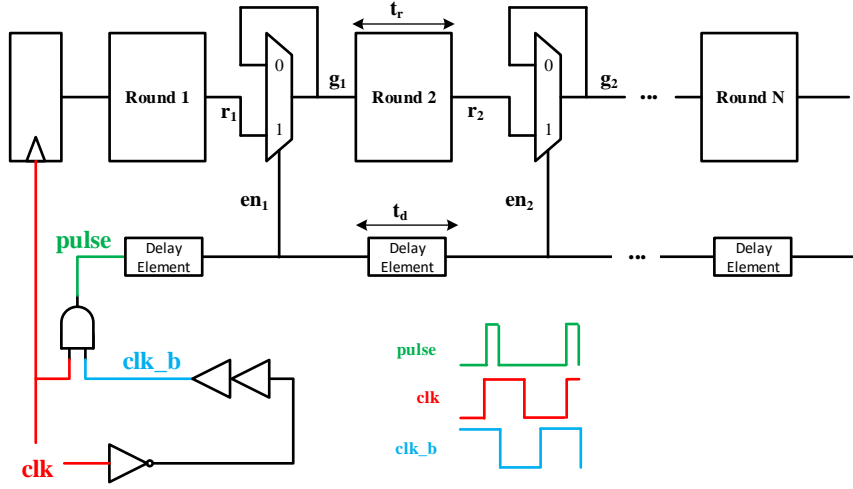


Figure 2.1: Schematic of latch-based checkpoints for glitch filtering.

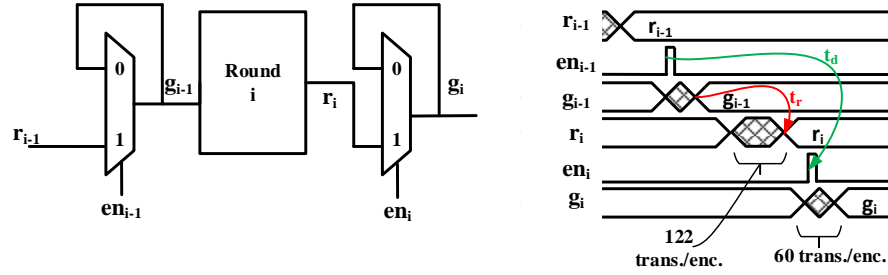


Figure 2.2: Timing diagram of glitch filter operation, annotated with the number of switching events happening at each point in the circuit for SIMON-128.

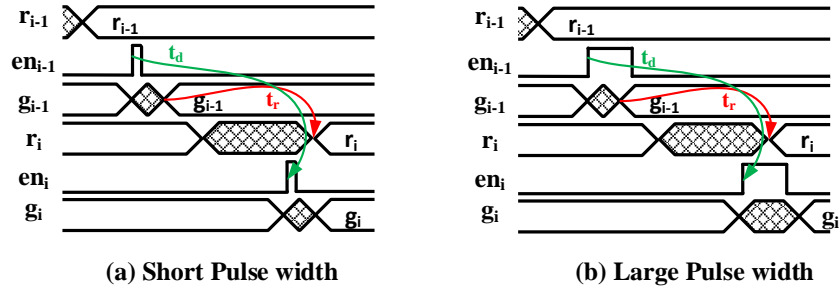


Figure 2.3: Increasing pulse width of enable signal to tolerate variations.

2.3 Evaluation of Checkpointing

2.3.1 Methodology

We use the SIMON and AES block ciphers to study the effectiveness of our glitch filtering scheme. SIMON is a lightweight Feistel cipher suitable for resource constrained systems, and we use SIMON-128 [13], which has a 128-bit key, 128-bit block size, and requires 68 rounds for each encryption. AES refers to three standardized variants [91] of the Rijndael cipher, based on a substitution-permutation network. Relative to SIMON, AES is a more complicated design, and we specifically use the widely used variant, AES-128; which has 128-bit block size, a 128-bit key, and requires 10 rounds per encryption. The RTL for both designs are written by us and validated for correctness against software implementations. To give an idea of the relative scales of the two ciphers, the round and key functions of fully unrolled SIMON require around 30,000 gates, whereas the round and key functions of fully unrolled AES are 4 times larger, requiring around 122,000 gates.

All of the measurements presented in this section are from simulation. Specifically, we simulate designs with 45nm NCSU PDK [2] implemented using CMOS logic style. Synopsys Design Compiler and HSIM are used for synthesis and circuit simulation, respectively. We rely on circuit simulation rather than power simulations using characterized libraries to ensure that glitch propagation effects are accurately captured. Given the time consuming nature of circuit simulation on large designs, which takes several days per encryption for the unrolled AES design, we simulate only two encryptions per design, using inputs that are chosen at random. The first encryption initializes the circuit state, and the second encryption is used for measuring metrics described below. The accuracy of our results should not be compromised by the small number of encryptions simulated because a block cipher’s behavior is fairly independent of the input value used. In support of this claim, the energy consumption of partially unrolled (17 rounds) SIMON for 100 random input vectors is shown in Fig.

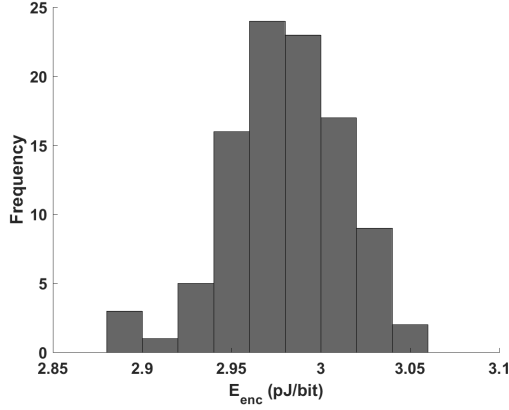


Figure 2.4: SIMON-128 energy per encryption histogram for 100 random inputs.

2.4. The variation in energy consumption is small ($\sigma = 0.032pJ/bit$) for the chosen input vectors.

Metrics such as toggle rate and energy consumption are measured during the circuit simulation and used to compare our scheme’s performance with others. **Toggle rate** is measured as the average number of signal transitions at round outputs per encryption. For example, in SIMON-128 a round output has 128 signals. We compute the total number of signal transitions in all 128 signals that occur during an encryption operation, and divide by bit-width (128) to get the toggle rate. We present energy numbers using a metric of **energy-per-encrypted bit** denoted as E_{enc} , which is the total energy consumed to perform an encryption operation divided by the number of encrypted bits generated during the operation. When considering individual rounds of the block cipher, we use as a metric the contribution of that round to the overall E_{enc} . In our experiments, clock frequencies are chosen such that idle time is minimal, and are above 10MHz in all cases.

2.3.2 Comparison of Average Switching Rates

We first study the effectiveness of the proposed glitch filter by counting switching events on a fully unrolled implementation of SIMON-128. Fig. 2.5 compares signal

toggle rates (signal transitions/encryption) for the outputs of all 68 rounds of SIMON. In the ideal case of no glitching activity, at the round outputs one can expect 0.5 transitions per signal for each encryption, as round outputs are uncorrelated across encryptions.

When no glitch filtering is used (baseline design), the switching activity is observed to increase linearly with logic depth (number of rounds). This increase in switching occurs because the logic of the block cipher tends not to mask transitions as they propagate, and because the diffusion property of block ciphers tends to propagate each transition out to many nodes. Our finding of linear increase is consistent with observations made in previous works [10]. For each encryption in the baseline design, the average switching across all rounds is 14.16 transitions per signal, and in the later rounds it is 2x larger than this average.

We analyze the effectiveness of checkpointing and two other techniques that mitigate switching. Compared to baseline, the Round Gating scheme [9] achieves a much lower average switching of 1.79 transitions per signal. Also, the switching activity stays fairly constant across rounds because glitches are never propagated across round boundaries. However as noted in Sec. 2.1, resetting the AND gates every clock cycle leads to unnecessary switching activity. Our checkpointing scheme has no such resetting and is therefore able to reduce switching to 0.95 transitions per signal, a 47% reduction relative to Round Gating. For comparison purposes, we implement SIMON-128 also using WDDL logic style [106]. WDDL is a dual-rail precharge based logic that is glitch free by design. To mitigate power side channel leakages, every signal pair in WDDL always has exactly 2 transitions per encryption; specifically, among the true and complementary representations of each signal, it is always the case that exactly one representation goes through a 1-0 transition during precharge and a subsequent 0-1 transition during evaluation.

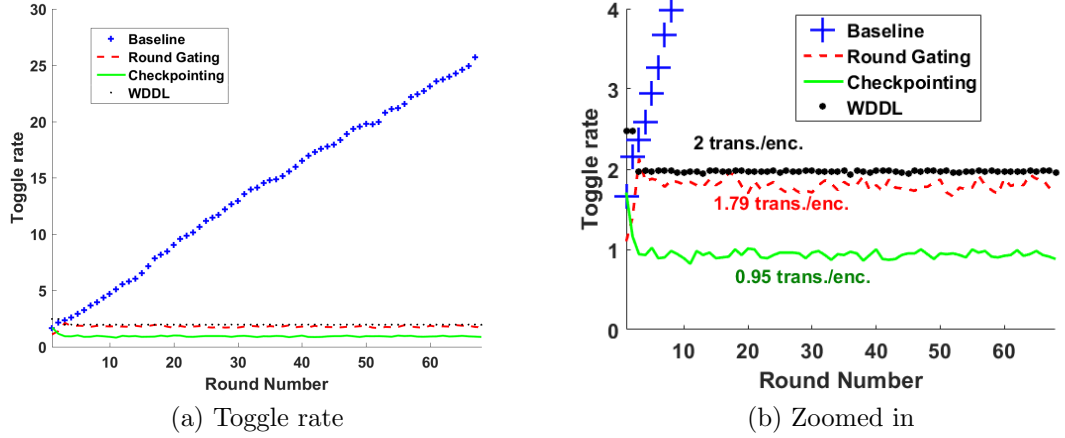


Figure 2.5: Comparison of the average toggle rate of the output signals of each round of SIMON-128 for four different implementation styles.

2.3.3 Energy Comparison in Fully Unrolled Designs

The significant reduction in average switching rates implies that glitch filtering can reduce the overall energy used for encryption. In this section we study the energy savings achieved by using checkpointing to filter glitches in fully unrolled implementations of SIMON-128 and AES-128.

2.3.3.1 SIMON-128

The energy use of each of the 68 rounds in the fully unrolled SIMON-128 implementation is plotted in Fig. 2.6 for the baseline (no glitch filter) design and three glitch filtering schemes. The energy trends across rounds are similar to the toggle rate trends shown in Fig. 2.5. The total energy per encryption (E_{enc}) including all of the rounds is given in Tab. 2.1 and is broken down by function to show where the energy is being used. A fully-unrolled implementation with checkpointing (4.46pJ/bit) is more efficient than fully unrolled baseline (25.91pJ/bit) because it greatly reduces the amount of energy spent on switching in the data and key rounds, and this savings is considerably larger than the energy spent to implement the checkpoints. In comparison to Round Gating [9], checkpointing consumes 27.9% lower E_{enc} . The

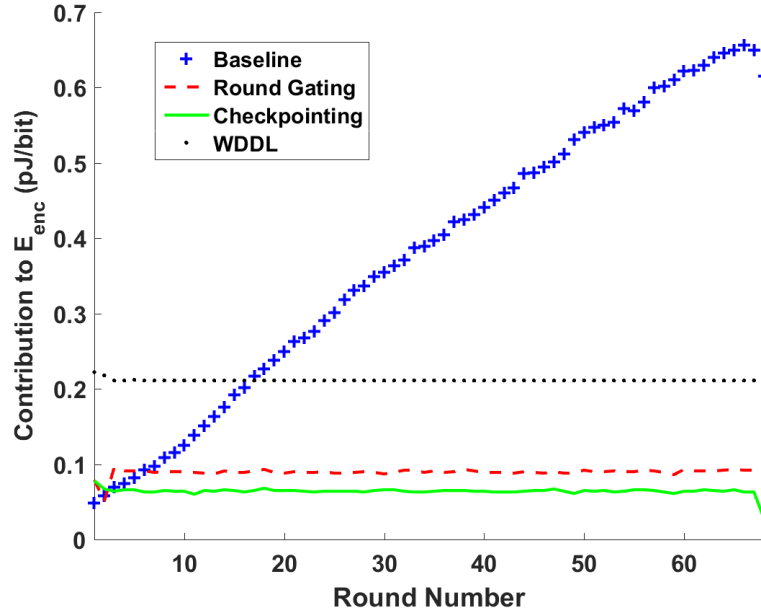


Figure 2.6: Contribution of each round to the overall energy per encrypted bit in four different implementation styles of fully unrolled SIMON-128

savings comes from a 47% reduction in toggle rate which leads to a 44.6% reduction in data and key computation energy specifically, while the costs of other components are similar across the two schemes. Note that WDDL and Round Gating schemes have similar toggle rates, yet WDDL consumes 2.4x more energy because it uses only positive gates, and therefore requires approximately 3x more gates to implement the same function.

Fig. 2.7a shows the breakdown of energy consumption per encryption for the checkpointing scheme. As can be seen in the figure, the switching energy does not increase across rounds, because each round similarly starts its computation from a single switching event. However, as was noted in Tab. 2.1, the glitch filters themselves consume about 50% of the total energy relative to the extremely simple combinational round function of SIMON. Hence, there is a possibility that using fewer glitch filters might reduce E_{enc} further if the glitches do not increase significantly. We explore this in Sec. 2.3.4. It can also be noted that the simple delay line that propagates

Table 2.1: Breakdown of E_{enc} (pJ/bit) in fully unrolled SIMON-128. Glitch filters are added after every round in Round Gating and our checkpointing work

SIMON-128	Baseline	Round Gating	Checkpointing	WDDL
Data	16.37	1.90	1.07	6.95
Key	9.42	1.62	0.88	7.42
Glitch Filter	–	2.36	2.20	–
Delay Line	–	0.18	0.19	–
Other	0.12	0.12	0.12	0.45
Total	25.91	6.19	4.46	14.82

the enable is not costly in energy, as it is a single inverter chain relative to a 128-bit wide computation path. The delay line does not require any tuning if care is taken by adding some margin (buffers) to ensure $t_d > t_r$ (Fig. 2.2) even in the presence of process variation.

2.3.3.2 AES-128

We repeat the energy analysis of checkpointing for the larger design, the fully unrolled implementation of AES-128. The energy breakdown per encryption in Fig. 2.7b shows that glitches are filtered effectively as there is no significant increase in switching energy with logic depth (round number). The energy cost of glitch filtering is small compared to that of actual computation. Note that the last round in AES is simpler, and therefore consumes less energy. The energy breakdown summary is tabulated in Tab. 2.2. Our scheme consumes an E_{enc} of 2.16 pJ/bit, which is 4.6x lower than fully unrolled baseline and 45.6% lower than Round Gating. These savings directly come from a lower switching activity. Unlike the extremely simple round functions of SIMON, AES round and key functions constitute more than 80% of the total energy. As a result, in comparison to Round Gating our scheme saves more energy in AES-128 (45.6%) than in SIMON-128 (27.9%).

Table 2.2: Breakdown of E_{enc} (pJ/bit) in fully unrolled AES-128. Glitch filters are added after every round in Round Gating and our checkpointing work

AES-128	Baseline	Round Gating	Checkpointing
Data	7.97	2.83	1.44
Key	1.83	0.76	0.36
Glitch Filter	—	0.30	0.26
Delay Line	—	0.02	0.04
Other	0.05	0.05	0.07
Total	9.85	3.97	2.16

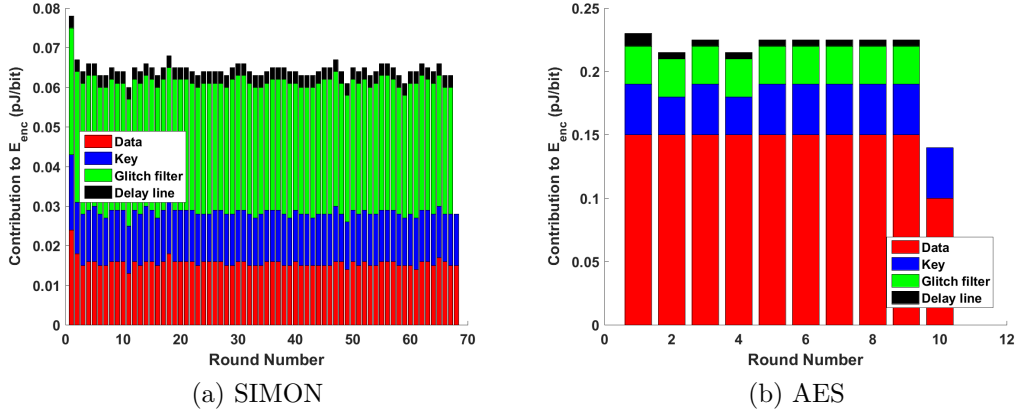


Figure 2.7: Energy/encryption breakdown in fully unrolled implementations using checkpointing after every round.

2.3.4 Optimal Placement of Checkpoints for Glitch Filtering

In this section we explore the optimal number of glitch filters that should be used to minimize the total energy consumption. Energy optimal glitch filtering requires finding the right trade-off between the cost of glitch filtering and the energy saved by filtering glitches. If too many filters are used, then the cost of the filters themselves will dominate; but if too few filters are used, then the cost of the glitches will dominate. Fig. 2.8 shows how each round contributes to the energy per encrypted bit when different numbers of rounds are implemented between the checkpoints. When

checkpoints are added after every round (spacing = 1) in fully unrolled SIMON-128 (Fig. 2.8a), more energy is spent in glitch filtering than is spent in actual computation. However, if checkpointing is done every other round (spacing = 2), the average energy per round is decreased because the reduction in glitch filtering energy is larger than the increase in switching energy of the key and data rounds. Increasing the spacing beyond 2 further reduces the cost of glitch filtering but the glitches increase the key and data energy by a larger amount and the total energy increases. Therefore a spacing of 2 rounds between checkpoints is optimal for SIMON-128.

The energy breakdown of E_{enc} for each round of the fully unrolled SIMON-128 with optimal glitch filter placement is shown in Fig. 2.9. Checkpoints are added after every second round - the even rounds have more glitching, and only the even rounds spend energy on checkpointing. At the optimal spacing of 2, the design consumes 4.18pJ/bit per encryption which is 6.3% lower than the 4.46pJ/bit when checkpointing is applied after every round (Tab. 2.1). In addition, the area will be reduced because of the fewer checkpoints. Any block cipher implementation will have some optimal tradeoff of checkpointing energy versus glitching, but the specifics are of course design and technology dependent.

Fig. 2.8b shows that in AES, the much larger round function justifies adding glitch filtering after every round. Therefore, checkpoints are added at all round boundaries in AES-128. Our design uses the Decode-Switch-Encode S-box implementation [17], which by design is not prone to glitching, so it is likely the energy penalty of going from 1 round spacing to 2 round spacing might be greater if a different S-box implementation were used.

2.3.5 Checkpointing in Partially Unrolled Designs

Partially unrolled designs, which implement some number of rounds combinationally, offer a tradeoff between area and latency of encryption. Aside from this tradeoff,

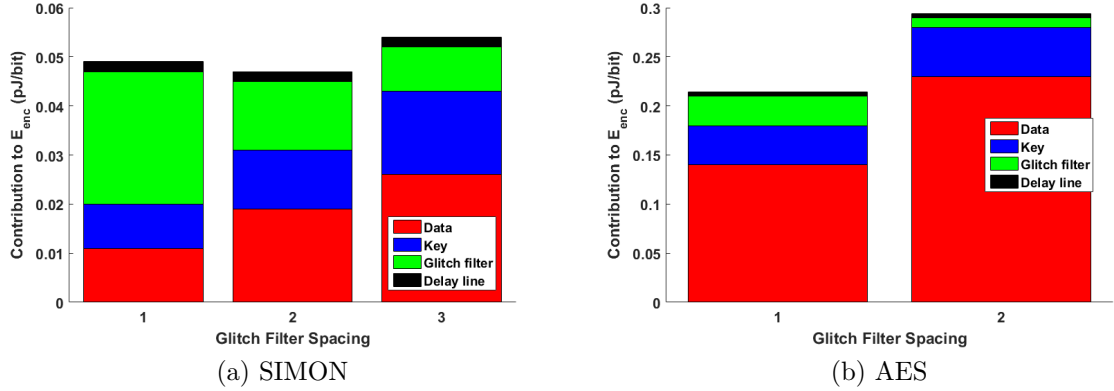


Figure 2.8: Energy efficiency varies with the spacing between checkpoints in fully unrolled designs. Performing more computation between checkpoints reduces checkpointing energy, but allows more data switching to occur

partial unrolling may also be desirable due to design constraints (area, clock period) which do not allow for a fully unrolled implementation. Since the optimal spacing of checkpoints is a low number (every round for AES-128, and every second round for SIMON-128), it is beneficial to use checkpointing even for partially unrolled designs.

2.3.5.1 SIMON-128

Tab. 2.3 shows the energy per encryption numbers for different partially unrolled implementations of SIMON-128. Glitching causes the energy of the baseline design to increase with the degree of unrolling up to 25.91 pJ/bit for the fully unrolled design. The energy savings offered by checkpointing also increase with unrolling up to 84% in the fully unrolled case. Checkpointing allows for a deeper unrolling while keeping the energy efficiency nearly constant. In comparison to the most efficient baseline implementation (4-unrolled, 2.89pJ/bit, 17-cycle latency), checkpointing enables 34-unrolled design (3.41pJ/bit, 2-cycle latency) to be competitive in energy at a much lower latency. Unrolling the design further (68-unrollings) helps save some loop control energy but incurs significant leakage cost, leading to a less efficient design (4.18 pJ/bit).

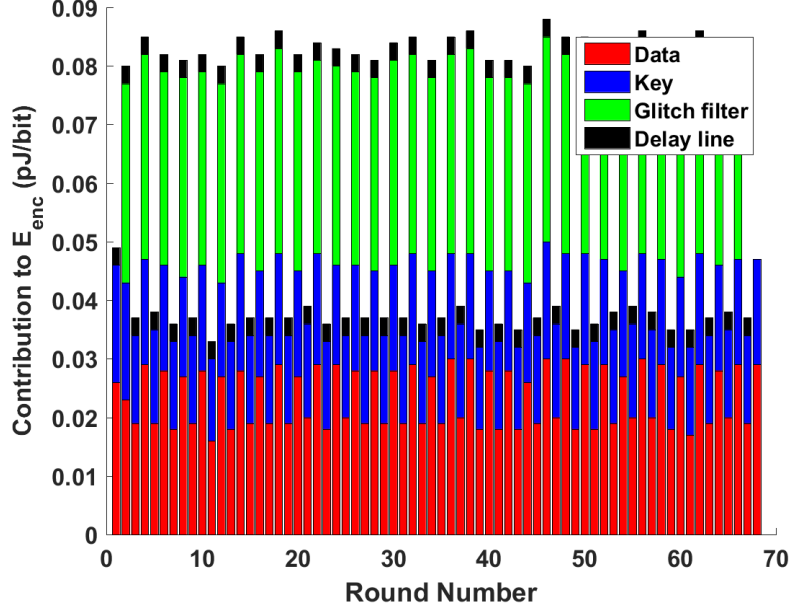


Figure 2.9: Energy breakdown of E_{enc} for each round in fully unrolled SIMON-128 in the optimal configuration of checkpointing every second round.

The 1-unrolled baseline design consumes more energy than the 2-unrolled and 4-unrolled baselines because the SIMON key expansion function requires storing key_{i-2} in additional registers to compute key_i if no unrolling were done [13]. The frequencies in Tab. 2.3 are chosen conservatively to account for process variations, but the design could be optimized for performance.

2.3.5.2 AES-128

Similar results for different partially unrolled implementations of AES-128 are tabulated in Tab. 2.4. The fully serialized Baseline (1.79pJ/bit) is the most energy efficient design. In comparison, checkpointing allows unrolled implementations to be competitive in energy while offering latency improvements. For example, the 2-unrolled checkpointed design (1.83 pJ/bit) reduces latency by a factor of 2 for a small 2.2% energy penalty with respect to the fully serialized Baseline. The cost of leakage

Table 2.3: E_{enc} (pJ/bit) comparison in SIMON-128 between optimal checkpointing and the baseline design for various degrees of unrolling.

Either	Unrolled rounds	1	2	4	17	34	68
	Latency (cycles)	68	34	17	4	2	1

Baseline	E_{enc} (pJ/bit)	3.78	2.95	2.89	6.15	12.43	25.91
	$I_{leak}(\mu A)$	133	134	170	417	753	1,420
	Frequency (MHz)	1,667	833	417	98	49	25

Checkpointing	E_{enc} (pJ/bit)	–	–	2.92	2.99	3.41	4.18
	$I_{leak}(\mu A)$	–	–	170	557	1,080	2,017
	Frequency (MHz)	–	–	185	73	37	19

increases with unrolling depth (Tab. 2.4) resulting in energy inefficiencies. Power gating can help in mitigating this problem as discussed in Sec. 2.3.7.

With regard to timing, unrolled designs operate at slow clock frequencies. Checkpointing incurs a small timing penalty because of the introduction of the glitch filters in the critical path and some timing margin to make sure the delay element is sufficiently long so that the enable pulse to a glitch filter arrives after the corresponding round output stabilizes. Though we report conservative frequency numbers in Tab. 2.4 to account for process variations, there is no requirement to double the (already slow) clock period as in other schemes such as WDDL or Round Gating.

Our colleagues at UMass evaluated the Checkpointing scheme on Xilinx and Altera FPGAs using unrolled implementations of SIMON and AES, and found similar energy savings [35].

2.3.6 Area Cost of Checkpointing

Using our glitch filtering scheme does incur some area penalty as tabulated in Tab. 2.5. In terms of number of gate equivalents, the area overhead is 4.2% if checkpoints

Table 2.4: E_{enc} (pJ/bit) comparison in AES-128 between optimal checkpointing and the baseline design for various degrees of unrolling.

Either	Unrolled rounds	1	2	5	10
	Latency (cycles)	10	5	2	1

Baseline	E_{enc} (pJ/bit)	1.79	2.84	6.16	9.85
	$I_{leak}(\mu A)$	560	1,030	2,400	4,600
	Frequency (MHz)	1,000	625	313	164

Checkpointing	E_{enc} (pJ/bit)	–	1.83	2.06	2.16
	$I_{leak}(\mu A)$	–	1,050	2,500	4,800
	Frequency (MHz)	–	500	159	81

Table 2.5: Area penalty of proposed glitch filtering scheme in units of gate equivalents. Even in absolute terms, the area cost of checkpointing is significantly higher in SIMON-128 than in AES-128 because the larger number of rounds requires a larger number of checkpoints, even though the checkpoints are only applied at every second round.

	Baseline	Checkpointing	Area overhead
SIMON-128	56,488	81,321	44.0%
AES-128	147,333	153,528	4.2%

are added after every round in AES-128. In the case of a lightweight block cipher like SIMON-128 that has a very small round function and larger number of rounds, the penalty is more pronounced. In SIMON, the area overhead is 44% if checkpoints are placed at the energy-optimal spacing of every second round. Adding checkpoints after every round in SIMON-128 would incur a much higher 80% area penalty in addition to not being energy optimal.

2.3.7 Power Gating

In unrolled designs of a block cipher, energy efficiency would be independent of unrolling depth except for leakage, which causes a linear increase in energy with unrolling when a design is run at its maximum frequency. Power gating idle round blocks can help reduce leakage power consumption. The round functions can be powered using a virtual supply, like the drain terminal of a PMOS header as shown in Fig. 2.10a. When the header is turned ON, the round functions are connected to the supply (V_{DD}) and can perform normal computation. When a round is idle, it can be turned off by simply turning off the header transistor to disconnect the power supply. The checkpoints are not power gated and remain powered using V_{DD} to ensure that state is retained. This isolates the checkpoints from the round functions, and any decay within the logic of the round functions will not cause additional power to be drawn from the supply.

Round functions between two checkpoints constitute a Power Gated Block (PGB), and each such block is connected to its own virtual supply with a dedicated header. In SIMON two rounds constitute a PGB as checkpoints are inserted every second round, whereas in AES each round is a PGB by itself as checkpoints are inserted after every round. When the checkpoint preceding a PGB is updated with new data, the round functions in the PGB need to be turned ON to compute on the new data. The PGB can then be powered off *after* the computed data is latched on to the checkpoint succeeding the PGB. To accomplish this, the control signal for a PGB's header is generated using an SR latch that is "set" by the enable signal of the preceding checkpoint (en_{i-2} in Fig. 2.10a) and "reset" by the enable signal of the succeeding checkpoint (en_i). The SR latch outputs a one when the set signal (en_{i-2}) is asserted, and stays high until the reset signal (en_i) is asserted. Thus a PGB is turned ON only while it is computing on new data.

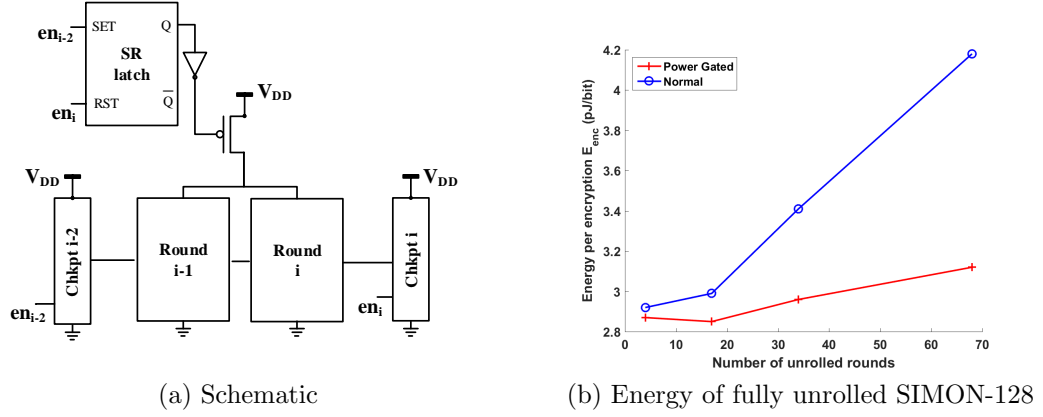


Figure 2.10: Power gating to reduce leakage power consumption.

The header transistor is sized so that it is able to supply the peak current required by the connected round functions. Each PGB's header is turned ON and OFF once per cycle, and from our experiments the energy spent in switching the header is smaller than leakage energy saved in the round functions. This leads to a net energy savings as seen in Fig. 2.10b, with up to 1pJ/bit being saved in the 68-unrolled case. Energy savings on using power gating for different partially unrolled implementations of SIMON-128 are shown in the figure.

2.3.8 Voltage Scaling

Supply voltage scaling is a well known general technique to reduce energy consumption of a circuit and has been applied to block ciphers as well [20, 78]. Switching energy reduces with supply voltage while the computation time increases. On scaling down the supply voltage beyond a certain point, leakage energy starts to increase and dominate the total energy consumption. Thus there is a minimum energy point of operation - which can be a near-threshold or even a subthreshold supply voltage depending on the design characteristics [23]. The effect of voltage scaling on E_{enc} of different partially unrolled implementations of SIMON-128 is shown in Fig. 2.11. The total energy consumption reduces with supply voltage and leakage energy (dotted

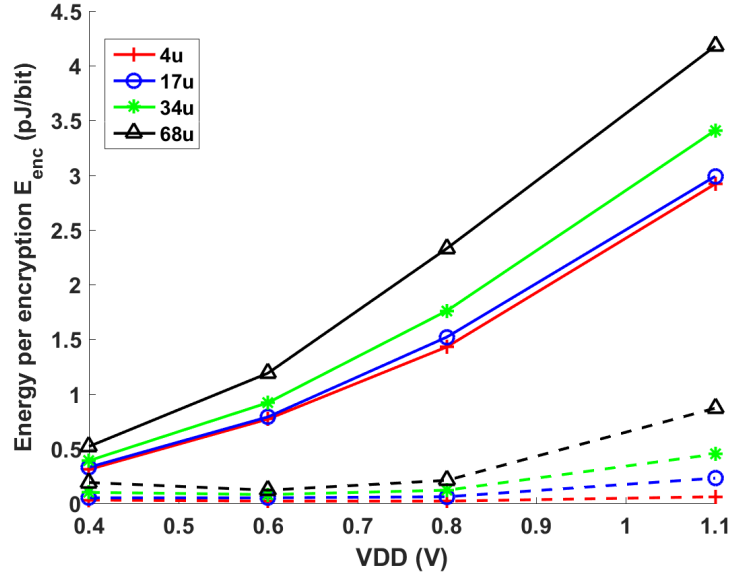


Figure 2.11: Energy per encryption at different supply voltages in SIMON. Dotted lines represent leakage energy.

lines) accounts for a major fraction of the total energy at low supply voltages. Regardless of the degree of unrolling, voltage scaling enables encryption to be performed at sub pJ/bit energy cost.

2.4 Summary

In this chapter, we have presented an efficient latch-based checkpointing mechanism to reduce the energy per encryption of unrolled block cipher implementations. We demonstrated significant energy savings (28-45%) compared to the best existing scheme for glitch filtering in unrolled block ciphers. Our scheme performs well on block ciphers with simple round functions as in SIMON, and complex round functions as in AES. We also showed that optimal use of glitch filters can result in energy consumption that is competitive to a fully serialized implementation while maintaining the latency advantages of an unrolled design. However unrolling comes at a significant

area cost, so we investigate lightweight block cipher implementations in subsequent chapters of this dissertation.

CHAPTER 3

EFFICIENCY IN LIGHTWEIGHT AES

Unrolled implementations of block ciphers as shown in the previous chapter have large area footprints, making them unsuitable for resource-constrained applications like IoT. Lightweight implementations take the opposite approach where less than a round is implemented in combinational logic. These sub-round implementations exploit the symmetric nature of operations in the round function, and save area by reusing fewer hardware units over time. For example in a round based implementation of AES (Fig. 1.1) the entire round function is performed combinatorially in one clock cycle. All 16 identical S-box instances are implemented in hardware and form a major (75%) contributor of area. Sub-round implementations of AES perform a fraction of a round in each clock cycle, and this allows a smaller number of S-boxes to be reused across clock cycles thereby saving area but increasing the number of clock cycles.

Compact AES implementations often use 8-bit data paths. In such a design, a single S-box circuit is reused 16 times per round, and therefore each round requires at least 16 cycles to complete. 8-bit implementations of AES are less energy-efficient than full-round implementations, and the inefficiency is mainly in the control and data movement. Among the computations performed in a round, SubBytes operates on 8 bits, and AddRoundKey is a bitwise XOR; only MixColumns is natively performed on 32-bit inputs, but is known to have an efficient serialization [47] that takes 8-bit inputs in four consecutive cycles. A complicating factor in sub-round AES implementations is that the round computation produces output bytes in an order that differs from their input order. For example, as shown in Fig. 1.1, one quarter of the round

computation uses bytes B_0, B_5, B_{10}, B_{15} and produces output values that will become bytes B'_0, B'_1, B'_2, B'_3 for the next round. The round output bytes are produced in sequential order if the input bytes are read in the order $(B_0, B_5, B_{10}, B_{15}, B_4, B_9, B_{14}, B_3, B_8, B_{13}, B_2, B_7, B_{12}, B_1, B_6, B_{11})$; we denote this ordering as Shift Rows Order (SRO). The reordering of bytes by the computation causes a Write After Read (WAR) hazard. As the first Mix Column outputs B'_0, B'_1, B'_2, B'_3 are produced, they must be written to a location that will not overwrite the current values of B_1, B_2, B_3 which have not yet been used in the current round. Since computation itself can scale down to an 8-bit datapath, the inefficiency of 8-bit architectures arises from the costs of moving data around and avoiding hazards. Two dominant techniques for moving data through the computation are RAM and shift register-based schemes.

Early 8-bit AES designs [37, 56] used small RAM blocks to hold state, and control logic to generate addresses to read and write the RAM. Because data can be written to, and read from, arbitrary addresses, these techniques make it easy to avoid data hazards without increasing the amount of storage available. The latency is high in these techniques (534 and 1016 cycles per block respectively) as very little useful work is performed in each cycle. RAM-based techniques can be low in power, but relatively higher in energy because of the energy cost of reading and writing data to and from RAM in each cycle.

Shift register-based datapaths improve on RAM-based datapaths and are the most compact way to orchestrate data movement in 8-bit AES. Most of the control complexity is handled implicitly by the wiring, and data bytes proceed in lockstep through the S-box and MixColumns at appropriate times. This shift register-based approach is employed by recent low power implementations [47, 116, 78] and shown to perform well. Note that the shift-register implementation style causes every byte of the state to move at least 16 times per round (e.g. 20 shifts per round in [47]), and this can have significant energy cost which we will address in this chapter. The total latency of

a shift register-based 8-bit AES can be as low as 160 cycles [116], which is a significant improvement over the RAM-based scheme.

3.1 Novel architecture

In this section, we describe a clocking methodology that improves energy efficiency of sub-round AES implementations. 8-bit architectures proposed in literature ([78, 116]) spend a lot of energy in data movement. These architectures move data through at least 16 registers per AES round. Our scheme uses register renaming to avoid data hazards without having to store a duplicate copy of the state register. Further, movement of each data byte is limited to 5 registers per round, thereby saving clock and data energy.

3.1.1 Improved clocking

In sub-round implementations of AES, care should be taken that the state register is not corrupted by WAR hazard as discussed earlier. Adding a shadow register file [78] to store intermediate results solves the problem but doubles the area of state registers. Shift register based schemes [47, 116] avoid this area penalty by storing the duplicate copy in the shifting behavior of the datapath. However, such an approach has energy inefficiencies due to data movement and clock load. Consider the architecture shown in Fig. 3.1 which has a state register whose bytes are individually clocked into and out of the registers using enable signals with a timing as shown in Fig. 3.2. The byte in physical register P_i is passed through the Shift Rows Mux to the S-Box when $enB[i]$ is active. The register enable signals $enB[i]$ are generated such that bytes are read out in Shift Rows Order, and the round function operates on one byte per cycle. The computed results are written back to the state register on the negative edge of the word enable signal $enW[i]$. In this scheme, each byte in the state register is clocked once per round as opposed to 16 times/round in other schemes [116]. Further, data

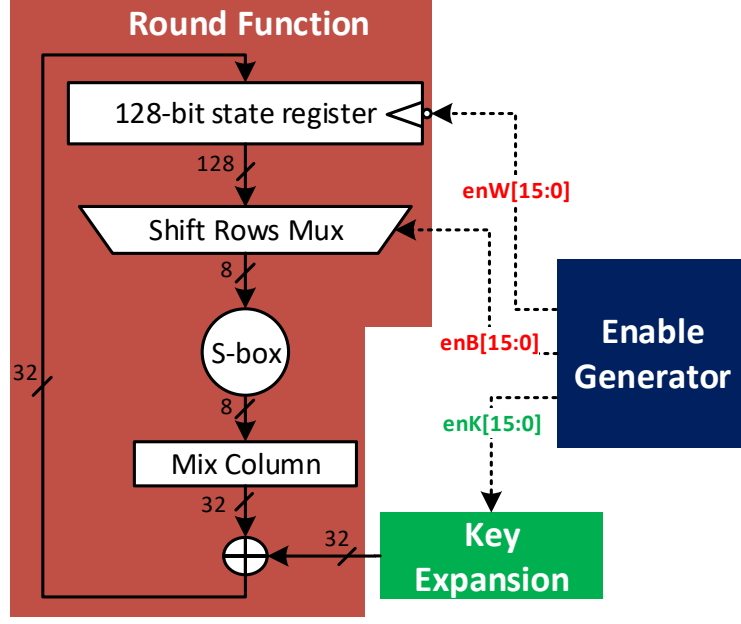


Figure 3.1: Proposed 8-bit architecture

moves through 1 state register and 4 registers in the Mix Column block which is again fewer than the 16 or more moves needed in shift register based schemes. One may notice that the proposed architecture (Fig. 3.1) does not contain any shadow registers. That is because a duplicate copy of the system state is not stored. WAR hazards are addressed in the following manner. Let byte B_i be read from register P_j for computation. Once the resulting output byte has been computed, it can be written back to register P_j as B_i is no longer required. However, the resulting byte is no longer byte B_i , so now the register P_j is logically renamed to ensure correct functionality.

3.1.2 Register renaming

Let P_0, P_1, \dots, P_{15} be 16 8-bit **physical** registers that store the 128-bit data. Similarly, let B_0, B_1, \dots, B_{15} be 16 8-bit **logical** registers, which also correspond to data bytes. The physical registers store the AES state, and the logical registers describe what byte is stored in each register. The correspondence between physical

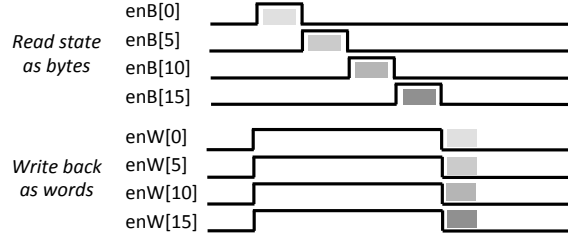


Figure 3.2: Timing of control signals for a quarter of one round. *enB* signals enable bytes to be read from physical registers into the datapath S-Box, and *enW* signals allow round outputs to be written back to physical registers on falling edge.

registers and logical registers changes over time, and a logical register may be found in different physical locations in different rounds of AES. At first appearance, this might seem to greatly complicate control flow, because a logical register required for the AES algorithm may need to be accessed from different physical registers across rounds. However, periodicity in register renaming results in a much simpler control logic as discussed below.

We first present the schedule of reading and writing registers used in our scheme, but we do not yet address the design of the control logic that generates the enable signals to realize this schedule. Once we’ve established here which physical addresses should be enabled in each cycle, we come back to the question of control logic in Sec. 3.1.3.

Fig. 3.3 illustrates our scheme of logically renaming registers to avoid the WAR hazards; each column in the figure corresponds to a physical address, and the markings in the squares denote the logical addresses contained therein during each cycle. Initially, the logical registers are mapped to the corresponding physical registers, that is $B_i = P_i$ for all i . The black squares in the figure indicate when data is read from each physical register, and the labels on those squares indicate which byte is stored in that register at the time of the read. The blue squares indicate cycles in which bytes of round output are written to physical registers, and the labels on the squares denote which bytes are being written to each register. Grey squares show the time between

writing a byte to a physical register and subsequently reading out that same byte. White squares indicate that the byte stored in the physical register has been read, but nothing has yet been written back. For example, in cycle 2 byte B_5 is read from register P_5 , computed on for two cycles and the resulting byte (B_1) is written back to P_5 in cycle 5, causing the register to be renamed accordingly. Round boundaries are indicated by thick lines (e.g. after cycle 16). Note that four bytes are written concurrently on every fourth cycle (i.e. in cycles 5,9,13,17 and so on). By the end of four entire rounds (64 cycles), all bytes are returned to the same physical registers in which they started, and the pattern repeats.

Note several very important details of Fig. 3.3. First, in each round, the bytes are read in Shift Rows Order ($B_0, B_5, B_{10} \dots$), although the pattern of reading from physical addresses that realizes this order changes across rounds due to the renaming. Second, in each round, the bytes are written in order with B_0, B_1, B_2, B_3 written first, then the next 4 bytes 4 cycles later, and so on. This means that, aside from the control logic that governs when each register is read and written, the remainder of the AES computation is entirely decoupled from the renaming and clocking scheme. The job of the control logic is then to read each of the physical registers at the times indicated by the black squares, and to write each of the physical registers at the times indicated by the blue squares.

At the beginning of the second round (cycles 17-21) bytes B_0, B_5, B_{10}, B_{15} , processed in Shift Rows Order, are read from physical registers P_0, P_9, P_2, P_{11} . The enable signals that control reading (writing) from (to) these physical registers are orchestrated by a control unit (Enable Generator in Fig. 3.1) that is aware of renaming and tracks bytes across physical registers. In the general case, in our scheme byte B_j , in round k , is mapped to physical register P_i , where i is as shown in Eq. 3.1.

$$i = (j + 12k(j \bmod 4)) \bmod 16 \quad (3.1)$$

Cycle	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
-	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
1	B0															
2						B5										
3											B10					
4																B15
5	B0				B4	B1					B2					B3
6										B9						
7															B14	
8				B3												
9				B7	B4				B8	B5					B6	
10														B13		
11			B2													
12								B7								
13			B10					B11	B8				B12	B9		
14		B1														
15							B6									
16												B11				
17	B0	B13					B14					B15	B12			
18										B5						
19			B10													
20												B15				
21	B0		B2		B4					B1		B3				
22														B9		
23							B14									
24																B3
25					B4		B6		B8					B5		B7
26		B13														
27											B2					
28				B7												
29		B9		B11					B8		B10		B12			
30						B1										
31															B6	
32								B11								
33	B0					B13		B15					B12		B14	
34														B5		
35											B10					
36								B15								
37	B0				B4			B3			B2			B1		
38		B9														
39															B14	
40												B3				
41		B5			B4				B8			B7			B6	
42						B13										
43			B2													
44																B7
45			B10			B9			B8				B12			B11
46										B1						
47							B6									
48				B11												
49	B0			B15			B14			B13			B12			
50		B5														
51			B10													
52				B15												
53	B0	B1	B2	B3	B4											
54						B9										
55							B14									
56								B3								
57					B4	B5	B6	B7	B8							
58										B13						
59											B2					
60												B7				
61									B8	B9	B10	B11	B12			
62														B1		
63															B6	
64																B11
-													B12	B13	B14	B15
Cycle	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15

Figure 3.3: Illustration of register renaming

Table 3.1: Physical registers to be enabled in each clock cycle

Round	Cycle	+0	+1	+2	+3
1,5,9	1	P_0	P_5	P_{10}	P_{15}
	5	P_4	P_9	P_{14}	P_3
	9	P_8	P_{13}	P_2	P_7
	13	P_{12}	P_1	P_6	P_{11}
2,6,10	17	P_0	P_9	P_2	P_{11}
	21	P_4	P_{13}	P_6	P_{15}
	25	P_8	P_1	P_{10}	P_3
	29	P_{12}	P_5	P_{14}	P_7
3,7	33	P_0	P_{13}	P_{10}	P_7
	37	P_4	P_1	P_{14}	P_{11}
	41	P_8	P_5	P_2	P_{15}
	45	P_{12}	P_9	P_6	P_3
0,4,8	49	P_0	P_1	P_2	P_3
	53	P_4	P_5	P_6	P_7
	57	P_8	P_9	P_{10}	P_{11}
	61	P_{12}	P_{13}	P_{14}	P_{15}

3.1.3 Implementation

In this section, we describe the implementation details of the architecture shown in Fig. 3.1. An AES round operation consists of Shift Rows (permute bytes from different words), substitution operation (S-box), followed by Mix Column (mix bytes from different words) and addition of round key. All these operations operate on bytes except the Mix Column which operates on words. The Enable Generator produces byte enable signals (enB in Fig. 3.1) for registers that cause the AES state to be passed through the Shift Rows MUX in Shift Rows Order during every round of encryption. The data then goes through the S-box, and gets mixed with three other bytes in the Mix Column block. Finally, 32 bits of the round key are added to the data and written back to the state register. As described in the previous section, data is written back to the register it was read from, and renaming ensures no data hazards occur.

Our enable generation logic allows bytes to be processed in appropriate order without the typical 8-bit architectural approach of shifting data through several flip-flops and multiplexers [54]. In our design, the outputs of the state register are multiplexed (by the Shift Rows Multiplexer - SRM) as shown in Fig. 3.1. We use the one-hot enB signals produced by the enable generator as select inputs to the 16:1 SRM. To preserve Shift Rows Ordering, the control circuitry generating enB needs to enable the physical registers in each cycle as listed in Tab. 3.1. The physical registers listed in the table correspond to the location of the black squares in Fig. 3.3 across four rounds. Note that each column in Tab. 3.1 can be described as a repeating pattern with a circular shift at round boundaries. The four columns have circular shifts of 0, 1, 2, and 3 positions at the round boundaries, respectively. This observation enables us to generate the 64 cycle pattern of control signals required for datapath orchestration and register renaming at the cost of just 23 single-bit registers, as discussed below.

Fig. 3.4 shows the Enable Generator. It consists of a single byte-select shift register and four word-select shift registers. As the name implies, the Word Select registers collectively enable a word (32 bits) that the Mix Column block operates on over 4 cycles. The Byte Select unit enables one byte of this word per clock cycle to pass through SRM and to use the datapath S-box (Fig. 3.1) before entering MixColumns. Each Word Select register shifts around a single 1 value and the current position of the 1 value determines which register is enabled in the current word. The position of the 1 within each word select register only changes on every fourth clock cycle (when *shift* is asserted). For example, the state of the Word Select registers shown in Fig. 3.4 causes enable signals $enW[0]$, $enW[5]$, $enW[10]$ and $enW[15]$ to be asserted for the next four cycles. This corresponds to the start of round 1 of AES, in which the first 4 bytes are read from registers P_0, P_5, P_{10}, P_{15} . The Byte Select unit sequences the enB signals for these registers to allow one of the 4 bytes byte per cycle to proceed

through SRM through the S-Box and into Mix Columns. Note that this sequence of enable signals across four cycles is the case shown in the waveforms of Fig. 3.2. At the end of the 4th cycle *rotate* is asserted, the Word Select shift registers all advance by one position, and the next word (P_4, P_9, P_{14}, P_3) is processed. Once a round of AES is completed at the end of 16 cycles, the control input to the multiplexers in the Word Select shift registers causes them to rotate, and P_0, P_9, P_2, P_{11} becomes the first word processed in the second round. This shift at the round boundary accounts for the register renaming, as these registers are the ones that contain bytes B_0, B_5, B_{10}, B_{15} (see cycles 17-20 of Fig. 3.3).

In this way, despite the apparent complexity of the control signals, the enable generation circuitry comprises only 23 flops. Effectively, the scheme works because the control logic is mimicing the AES shift rows structure, but doing so in the control logic to avoid moving entire bytes around the datapath. Note that the flops in the Word Select shift registers are clocked by the system clock and not a divided clock, even though they only shift every fourth cycle. This is done to avoid having an additional clk-to-q delay on the critical path, as would occur if the shift register used a derived clock.

The enable signals for the Key Expansion unit (*enK*) are computed by treating Word Select register WS_A and Byte Select register in Fig. 3.4 as word address and byte address respectively. *enK* selects each of the 16 key registers (Fig. 3.6) one per clock cycle in sequential order.

The schematic of the round function is shown in Fig. 3.5. All registers and data wires in the figure are 8-bits wide. The state registers are shown in red, and are organized in four groups $\{P_0, P_4, P_8, P_{12}\}$, $\{P_1, P_5, P_9, P_{13}\}$, $\{P_2, P_6, P_{10}, P_{14}\}$ and $\{P_3, P_7, P_{11}, P_{15}\}$. The inputs of all registers in a group are tied together, but since register P_i is clocked by negative edge of $enW[i]$ signal, a byte is always written to one register in a group and is ignored by the other three in the group because their

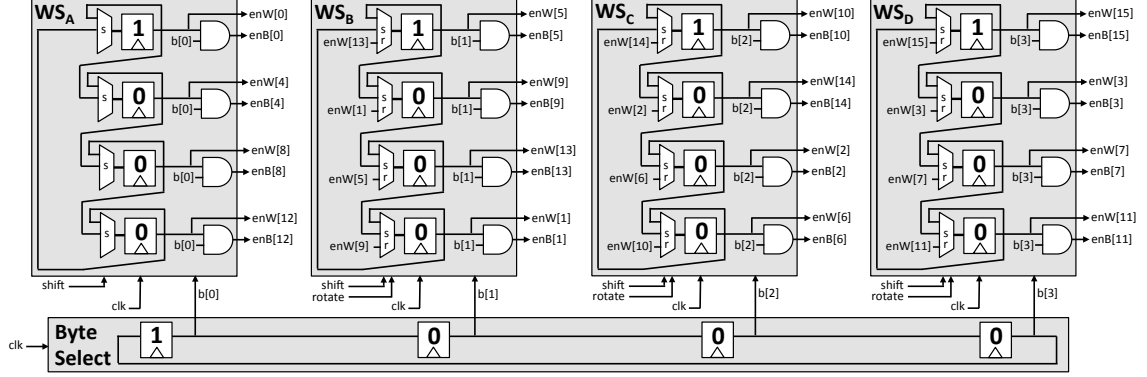


Figure 3.4: Enable Generator

clocks do not switch. The outputs of all 16 P_i registers are connected to the S-Box input via SRM with the byte enables $enB[i]$ acting as the respective select signals.

During a regular round computation, a four-byte word is selected by asserting four enW signals (Sec. 3.1.3) for a quarter of a round. enB signal then enables one byte of the word per cycle to pass through to the S-Box. We choose Decode Switch Encode S-box which performs one hot encoding to eliminate glitches and reduce energy consumption [17]. However, our architecture is agnostic to the choice of S-box and one can choose area efficient alternatives [78, 24] if desired. The S-box operation is followed by Mix Column operation which is performed over 4 clock cycles operating on the enabled word. We adopt the Mix Column design from [47] except that we do not pipeline the output. Instead, 32-bits are read out of MixColumns, XORed with 32 bits of round key and written into state at once; this decision prevents stalls, saves three register moves, and reduces clock loading. Given that enW is serving as the clock to the 128 bits of AES state, the registers in our design switch only once per round as opposed to once per cycle in conventional 8-bit architectures [47].

In AES with 128-bit key size, rounds 0 and 10 operate differently than the other rounds. In our scheme, rounds 0 and 10 work as follows. In round 0, plaintext bytes ($Data_{in}$) are read sequentially and XORed with corresponding input key bytes (Key_{in}). To match the “word write” of the regular round, we use three registers to

pipeline the data. These registers can be clock gated after round 0 to save energy. For round 10, Mix Column operation is not required and we XOR the S-Box output with 8-bits of round key to output an encrypted byte *Data.out* (Fig. 3.5). Note that since AES operation is decoupled from renaming, encrypted bytes are output in correct sequential order.

Each round of AES is completed in 16 cycles, leading to a latency of 160 cycles to encrypt a block. Among all the registers in our design, only the Enable Generation and the 32 bits of MixColumn state switch every clock cycle, which is a small percentage of the overall registers in the design. The state registers switch once per round thereby reducing clock load significantly. When considering that each data byte will be clocked through the state once and clocked 4 times through MixColumns, this adds up to only 5 register moves per byte per round, as opposed to approximately 20 moves in conventional 8-bit architectures [47, 78].

The schematic of Key Expansion is shown in Fig. 3.6. The functionality of key expansion is straightforward (register read/write is sequential) and interested readers can refer to [91] for more detail. The enable signals for the key registers (*enK*) are generated by the Enable Generator. These signals enable registers K_0 through K_{15} in the same sequence one register per cycle. The key registers are similar to data registers in that their outputs are multiplexed and they sample data on the negative edge of *enK*, once per round. The byte enables used as select signals are as shown in Fig. 3.6. Each byte i from words K_a, K_b and K_c is enabled twice, once while computing round key, and once more while being XORed with corresponding byte in the next successive word. Bytes from word K_d are also enabled twice - for computing g-function and round key.

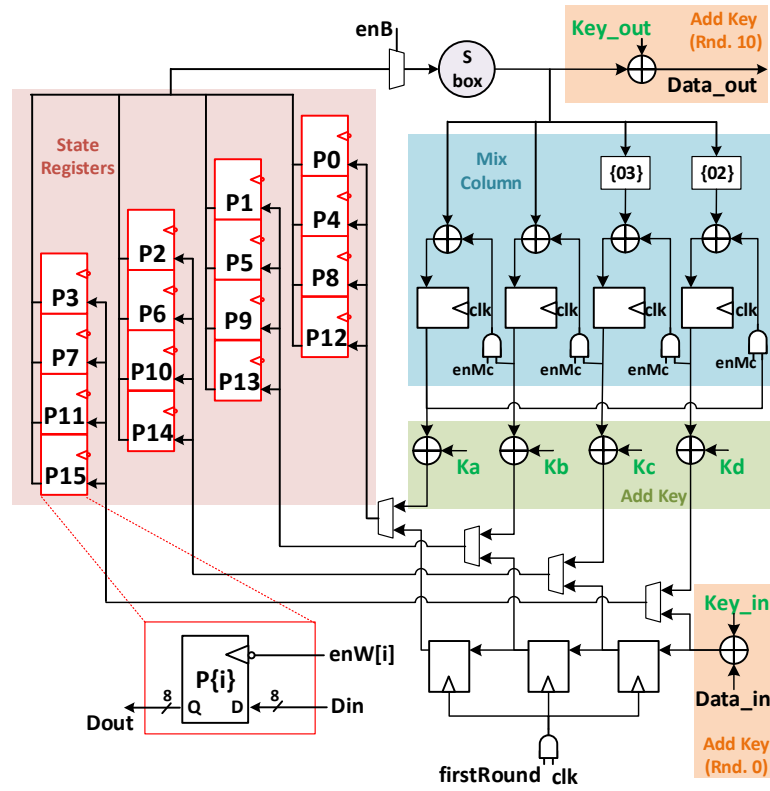


Figure 3.5: Schematic of Round Function

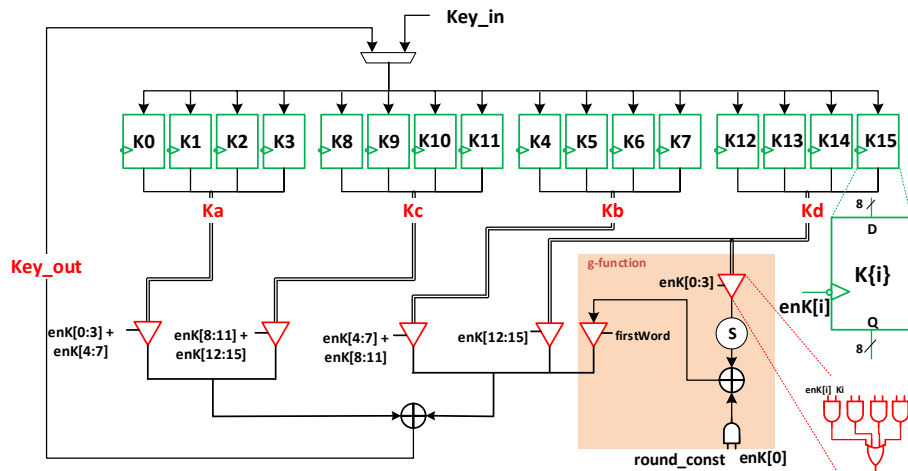


Figure 3.6: Schematic of Key Expansion

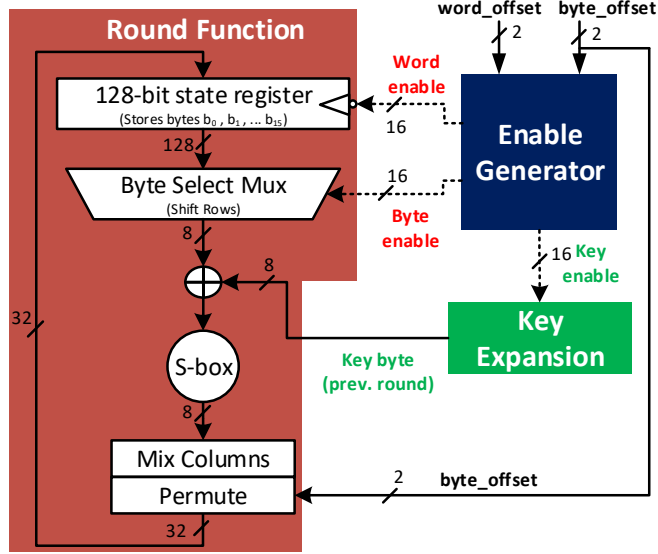


Figure 3.7: Randomization-capable 8-bit AES architecture. Additional details of redesigned Enable Generator and MixColumns circuits are shown in Figs. 3.9 and 3.10.

3.2 Microarchitectural Randomization

The security of the AES algorithm itself has held up against extensive scrutiny, but can often be broken by power side channel attacks (Sec. 1.3.1). In practice, it is often the case that an AES key can be guessed with a few thousand power measurements if side channel countermeasures are not used. Recent works have shown that sub-round architectures are especially susceptible to side channel attacks [100]. Randomizing the order of processing can potentially help resolve the side channel vulnerability, but requires architectural support as we will show. The renaming architecture from the previous section is modified to allow for processing the bytes in a randomly chosen order for every round (Fig. 3.7).

3.2.1 Enable Generator design for Word and Byte shuffling

The Enable Generator controls which bytes of the state register are read and written in each cycle of computation by generating the control signals to the byte selection mux and the state register itself. Within the Enable Generator these signals

come from four 4-bit Word Select Registers (WSRs) and a single 4-bit Byte Select Register (BSR). Fig. 3.9 shows the details of BSR and one 4-bit WSR. The working of the Enable Generator without word/byte shuffling is as follows. Each WSR is initialized to a specific one-hot value, where the position of the 1 bit selects a byte of the state register that is used in the current word of computation; the four WSRs together select the four bytes comprising the current word. The BSR selects in each of the next four cycles which byte of the word is read from the state register, through the S-Box, and into MixColumns. After four cycles the current word is written back *in-place* to the state register, and a shift signal is asserted to advance the WSRs by one position to select the four bytes of the word that is computed in the next four cycles. The use of in-place write back to the state register avoids data hazards, but changes the assignment of bytes to registers and necessitates register renaming for tracking the bytes. The logical renaming of registers at the end of each round is handled by asserting the rotate signal of each WSR, which updates the state of each WSR in a way that accounts for the renaming. More details regarding the working of renaming can be found in Sec. 3.1.2.

Since bytes are multiplexed into the datapath, by modifying the Enable Generator circuit one can shuffle the order in which the four words of the round are processed. To implement word shuffling, in each WSR the nominal rotation that happens at the end of a round is modified to include a random offset determined by the 2-bit **word_offset** signal (Fig. 3.9). This effectively causes each round to use a random choice of the word that will be processed first, and the other three words follow it in sequence. With the word ordering shuffled, there are now four cycles of the round in which each byte could be read from the state register and processed. For example, b_0 would still be processed as the first byte of its word, but that processing might occur in cycles 0, 4, 8, or 12 of the round depending on the value of **word_offset**.

		Byte read from state register in each cycle of round															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value of word_offset, byte_offset in round	0,0	b0	b5	b10	b15	b4	b9	b14	b3	b8	b13	b2	b7	b12	b1	b6	b11
	0,1	b15	b0	b5	b10	b3	b4	b9	b14	b7	b8	b13	b2	b11	b12	b1	b6
	0,2	b10	b15	b0	b5	b14	b3	b4	b9	b2	b7	b8	b13	b6	b11	b12	b1
	0,3	b5	b10	b15	b0	b9	b14	b3	b4	b13	b2	b7	b8	b1	b6	b11	b12
	1,0	b12	b1	b6	b11	b0	b5	b10	b15	b4	b9	b14	b3	b8	b13	b2	b7
	1,1	b11	b12	b1	b6	b15	b0	b5	b10	b3	b4	b9	b14	b7	b8	b13	b2
	1,2	b6	b11	b12	b1	b10	b15	b0	b5	b14	b3	b4	b9	b2	b7	b8	b13
	1,3	b1	b6	b11	b12	b5	b10	b15	b0	b9	b14	b3	b4	b13	b2	b7	b8
	2,0	b8	b13	b2	b7	b12	b1	b6	b11	b0	b5	b10	b15	b4	b9	b14	b3
	2,1	b7	b8	b13	b2	b11	b12	b1	b6	b15	b0	b5	b10	b3	b4	b9	b14
	2,2	b2	b7	b8	b13	b6	b11	b12	b1	b10	b15	b0	b5	b14	b3	b4	b9
	2,3	b13	b2	b7	b8	b1	b6	b11	b12	b5	b10	b15	b0	b9	b14	b3	b4
	3,0	b4	b9	b14	b3	b8	b13	b2	b7	b12	b1	b6	b11	b0	b5	b10	b15
	3,1	b3	b4	b9	b14	b7	b8	b13	b2	b11	b12	b1	b6	b15	b0	b5	b10
	3,2	b14	b3	b4	b9	b2	b7	b8	b13	b6	b11	b12	b1	b10	b15	b0	b5
	3,3	b9	b14	b3	b4	b13	b2	b7	b8	b1	b6	b11	b12	b5	b10	b15	b0

Figure 3.8: Shuffled orders in which bytes can be processed in our architecture. Depending on the value of the word offset and byte offset, each of the 16 state bytes could be processed in any of the 16 cycles of the round.

Ensuring that each byte can be processed in any of the 16 cycles of a round further requires shuffling of bytes within each word in addition to the shuffling of the words. Byte shuffling uses a similar mechanism to word shuffling. Before the start of each round, the BSR logic gets initialized to a state determined by the 2-bit `byte_offset` signal (Fig. 3.9). Within the word selected by the WSRs, the BSR will therefore cause the processing to start from a randomly chosen byte of the word. The remaining three bytes of the word follow in sequence before the next word is processed. Thus, word and byte level shuffling allow each data byte to be processed in any of the 16 cycles of a round, as seen in Fig. 3.8.

3.2.2 Mix Columns design to handle permutation

Among the AES operations ShiftRows, SubBytes (S-Box) and key addition require no changes to accommodate byte shuffling, whereas MixColumns requires modification. The MixColumns operation is performed on 4 bytes (a word) and is sensitive to the order of operated bytes. The function computed by MixColumns is shown

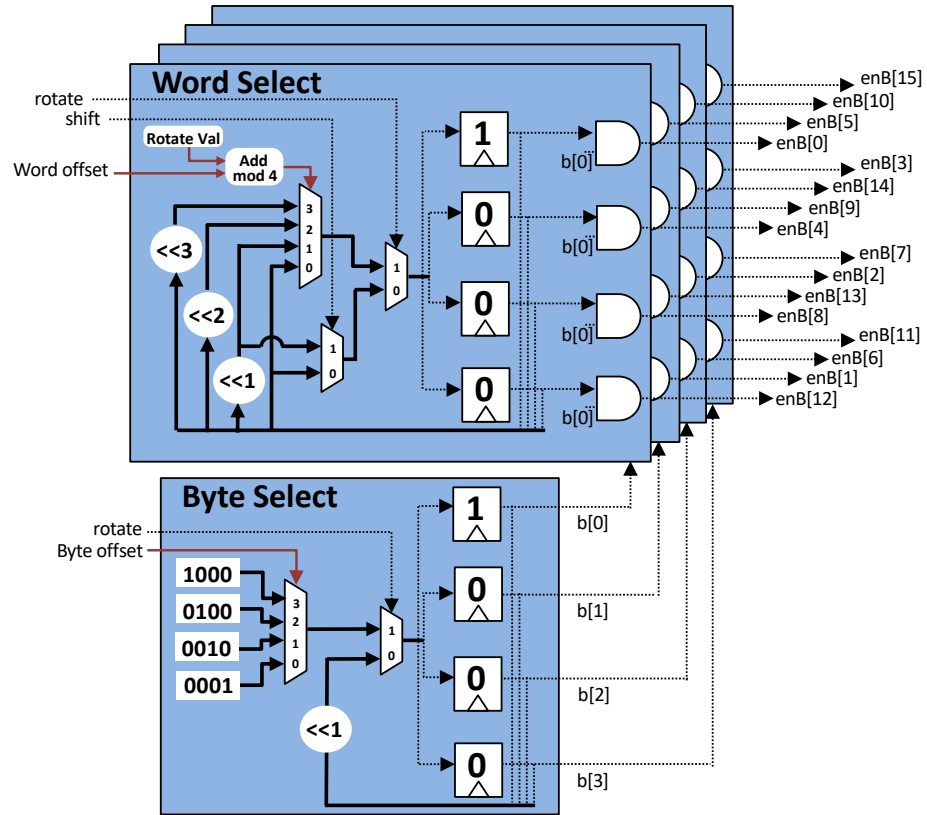


Figure 3.9: Enable Generator for Randomized architecture (compare Fig. 3.4). Solid, dotted and red lines indicate 4-bit, 1-bit and 2-bit signals respectively.

Table 3.2: Table illustrates the operation of the pipelined MixColumns (see Fig. 3.10) for two different orderings of the input bytes. At the end of the four cycles, the same values exist in the registers for both orderings, but their locations differ. The permutation step associates the appropriate register value to each output signal.

Cycle	Bytes arrive in the order s_0, s_1, s_2, s_3				Bytes arrive in the order s_1, s_2, s_3, s_0			
	Reg0	Reg1	Reg2	Reg3	Reg0	Reg1	Reg2	Reg3
1	s_0	s_0	$3s_0$	$2s_0$	s_1	s_1	$3s_1$	$2s_1$
2	s_1+s_0	s_1+3s_0	$3s_1+2s_0$	$2s_1+s_0$	s_2+s_1	s_2+3s_1	$3s_2+2s_1$	$2s_2+s_1$
3	$s_2+s_1+3s_0$	$s_2+3s_1+2s_0$	$3s_2+2s_1+s_0$	$2s_2+s_1+s_0$	$s_3+s_2+3s_1$	$s_3+3s_2+2s_1$	$3s_3+2s_2+s_1$	$2s_3+s_2+s_1$
4	$s_3+s_2+3s_1+2s_0$	$s_3+3s_2+2s_1+s_0$	$3s_3+2s_2+s_1+s_0$	$2s_3+s_2+s_1+3s_0$	$s_0+s_3+3s_2+2s_1$	$s_0+3s_3+2s_2+s_1$	$3s_0+2s_3+s_2+s_1$	$2s_0+s_3+s_2+3s_1$
Out	m_0	m_1	m_2	m_3	m_1	m_2	m_3	m_0

by Eq. 3.2, where s_i and m_i represent the i^{th} input and output bytes respectively of the current MixColumns word. In absence of shuffling, MixColumns can be efficiently serialized using a four stage pipeline [47] in which each incoming byte is scaled appropriately and accumulated to a four byte value (see Tab. 3.2).

Once byte shuffling is considered, the MixColumns pipeline that processes bytes s_0, s_1, s_2, s_3 may receive these bytes in four different orders: (s_0, s_1, s_2, s_3) , (s_1, s_2, s_3, s_0) , (s_2, s_3, s_0, s_1) , or (s_3, s_0, s_1, s_2) . If the standard MixColumns pipeline is used to process shuffled inputs, the correct output byte values will be computed, but their positions will be shuffled as is shown in Tab. 3.2. Our MixColumns design adds a new permutation stage that unshuffles the MixColumns results before they are written to the AES state register (Fig. 3.10).

$$\begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (3.2)$$

3.2.3 Sequencing of Rounds and Key Expansion

Although the sub-round computations are shuffled internally in the datapath, the external interface to the AES module remains unshuffled, meaning that the plaintext data is shifted into the state register in normal order, while the ciphertext data is

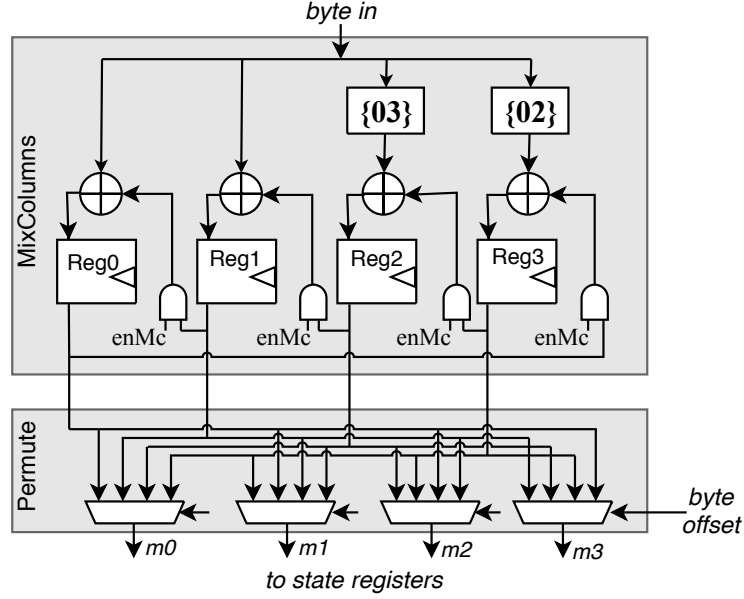


Figure 3.10: MixColumns for Randomized architecture.

concurrently shifted out of the state register in normal order. Shuffling is neither required nor possible while shifting plaintext/ciphertext as no computation is performed on the data at this time. After the data is loaded, the initial key addition is performed one (shuffled) byte at a time, and the 10 subsequent rounds of encryption are performed using the 8-bit datapath. There is an extra one-cycle delay at the end of each round to allow output of MixColumns to be written back to the state register before the next round begins. The one cycle delay avoids a potential read-after-write data hazard that can arise in shuffled operation when the same register is written at the end of one round and read in the first cycle of the next.

Key expansion cannot easily be randomized on-the-fly because its computation has a long chain of dependencies. Instead, the key addition which normally happens at the end of a round before data is written back to the state register, in our design is delayed to occur in the next round as the first step when data is read from the state register. Delaying the key addition ensures that the required bytes of the round key

will always be available when needed, regardless of order in which the data bytes are processed in each round.

3.3 Evaluation

We evaluate the efficiencies of four AES architectures: (1) an ordinary round-based 128-bit design, (2) our efficient 8-bit renaming architecture [32], (3) our 8-bit design with randomization, and (4) state-of-the-art reference 8-bit design [116]. The RTL for all designs are written by us and validated against an online tool. For a fair comparison, all designs are synthesized using Synopsys Design Compiler for the same commercial 16nm FinFET technology, and all designs use the same energy-efficient S-Box implementation [17]. We use the nominal voltage (0.8 V) for our experiments. In addition to energy efficiency, the designs are also evaluated for side channel vulnerability, area, power and performance costs.

3.3.1 Efficiency and overheads

We first analyze the energy expenditure of the AES designs using Synopsys PrimeTime power simulations. Tab. 3.3 compares the energy per encryption (fJ/bit) broken down by different design modules (S-box, Mix Column, Key Expansion, Control) and also by design components (CLK, Sequential, Combinational). For efficiency, it is generally understood that no narrow datapath computation can match or exceed the efficiency of a 128-bit datapath, because serializations perform all of the same computation, but with some additional work to orchestrate the serialized design and store intermediate results. The 128-bit datapath, as expected, is the most energy efficient design at 244 fJ/bit. Our implementation of the reference 8-bit design [116] consumes 1350 fJ/bit for an encryption.

Our renaming architecture at 710 fJ/bit achieves a 47% improvement over the 8-bit reference design. The energy numbers shown in Tab. 3.3 demonstrate the

Table 3.3: Comparison of energy efficiency of four AES designs all implemented by us in the same 16nm technology.

	128-bit Ref.	8-bit renaming	8-bit Rand.	8-bit Ref.
S-box (data&key)	68	86	74	141
Mix Column	36	172	191	187
Key Expansion	52	84	80	409
Control	3	121	385	24
Energy (fJ/bit)	244	710	889	1350
CLK	38	246	325	741
Sequential	21	155	230	228
Combinational	185	309	334	381

specific benefits of using our clocking methodology. From table in comparison to 8-bit reference, our renaming architecture consumes 32% less sequential energy because data moves through 5 flip-flops per round instead of 20. Further, our design spends 3x smaller CLK energy. This is because all 296 flops in the reference design switch every clock cycle. In our design 61 flops involved in Mix Column operation and enable generation switch every cycle, while the rest (280) of the flops holding data and key switch once per round.

Our randomization enabled 8-bit design at 889 fJ/bit uses 25% more energy-per-bit than the renaming design. Most of the additional energy is in the control logic which orchestrates and manages the shuffling of the data bytes, but the energy cost of MixColumns also grows due to the added permutation stage. Note that non-randomized reference 8-bit design consumes 1350 fJ/bit of energy, which is less efficient than our randomized design mainly due to inefficiency in data movement.

An area comparison is presented in Tab. 3.4. The 128-bit design is obviously the most expensive in terms of area. Our renaming design occupies $886 \mu m^2$ and incurs a 18% area penalty compared to our implementation of the reference 8-bit design [116]. This area penalty probably comes from additional multiplexers used as the control logic (Enable Generator) area is comparable to that of reference 8-bit. Randomization

Table 3.4: Comparison of area of four AES designs all implemented by us in the same 16nm technology.

	128-bit Ref.	8-bit renaming	8-bit Rand.	8-bit Ref.
S-box (data&key)	2226	223	223	223
Mix Column	217	60	85	60
Key Expansion	216	212	213	184
Other	407	391	476	104
Area (μm^2)	3066	886	1201	750

Table 3.5: Comparison of performance. Throughput obtained at 300MHz clock.

	128-bit Ref.	8-bit renaming	8-bit Rand.	8-bit Ref.
Energy (fJ/bit)	244	710	889	1350
Power (μW)	391	77	80	146
Area (μm^2)	3066	886	1201	750
Throughput (Mbps)	3491	239	187	239
Latency (cycles)	10	160	204	160
Mean MTD	9746	1492	15983	-

adds 35.5% area overhead on top of the 8-bit renaming design, as shown in Tab. 3.4. Both 8-bit designs are significantly smaller than the 128-bit datapath, owing primarily to their serialized use of a single S-Box instead of the 16 parallel S-Box instances in the 128-bit datapath. The area overhead of adding randomization to the 8-bit datapath comes primarily from designing the flexible Enable Generator and adding a permutation stage after MixColumns to allow reordering.

Finally, our designs are competitive in performance with the reference 8-bit design [116] as seen in Tab. 3.5. The renaming architecture offers similar throughput at 300MHz, while the randomized design has a 22% throughput degradation due to an increased latency. The latency increase is due to several factors: additional stall cycle at the end of each round, delayed key addition and reading out cipher text *after* final key addition to prevent information leakage.

3.3.2 Susceptibility to Side Channel Attacks

To provide a baseline comparison between the architectures, side channel attacks are performed on standard CMOS implementations of the designs in absence of other circuit-level countermeasures. Attacking baseline implementations gives a fair comparison of the relative susceptibility of each architecture, which is important to consider. If the architectures are strengthened by circuit-level countermeasures, the relative side channel susceptibilities of the architectures will reflect the baseline comparison. In the randomized design shuffling of data requires four random bits; 2 for the `word_offset` signal that selects the ordering of the words, and 2 for the `byte_offset` signal that selects the ordering of the four bytes within a word. We shuffle each round independently, and assume that the random bits that configure the shuffling are generated by the chip’s true-random number generator or cryptographically secure random number generator. Given that the shuffling bits are not externally observable, it might be permissible to shuffle less often, but we do not consider this.

We perform logic simulation to obtain switching activity information and subsequently use time-based power analysis in Synopsys PrimeTime to generate the power traces on which the attacks are performed. The differential power analysis attack is performed on the power traces using a program written in C++. The attack deployed against the designs is a standard DPA attack, so we provide limited details here and refer interested readers to the work of Kocher et al. [63], among others, for a description of DPA. Per usual in DPA, we assume that the attacker knows when the encryption begins and can use this to align the power traces. The selection function of the DPA attack is based on predicted values of an intermediate circuit node under each key byte guess. In our experiments we use an S-Box input bit as selection function during the final round of each encryption.

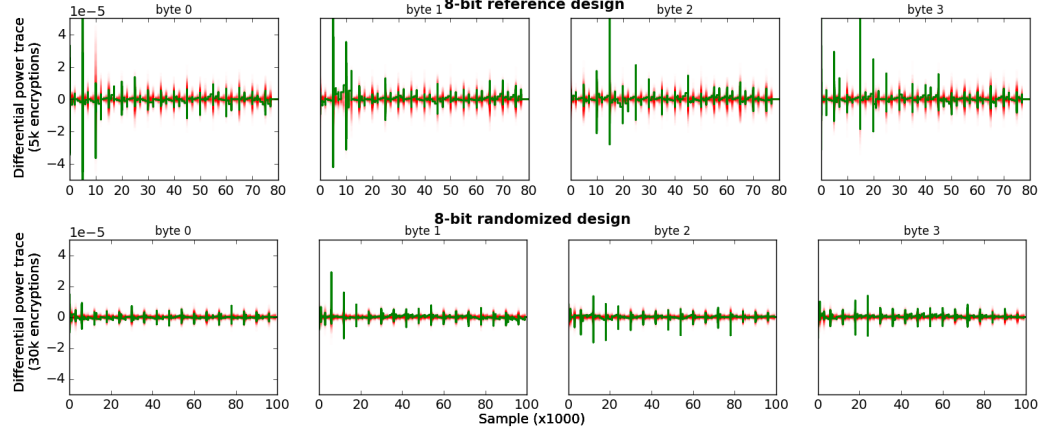


Figure 3.11: Plots showing DPA attack on 4 key bytes for the 8-bit designs. Top plot shows differential power traces for the 8-bit renaming design and the bottom plot shows the same for the 8-bit randomized design. Green line corresponds to correct key guess and red lines correspond to incorrect key guesses.

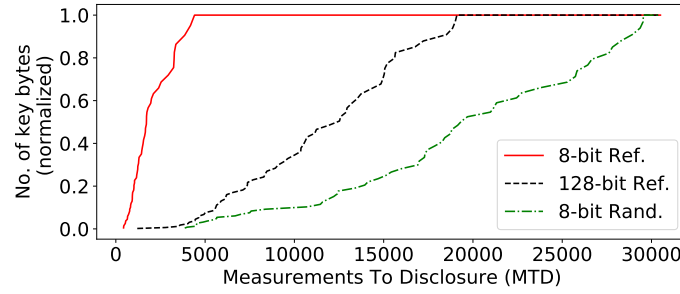


Figure 3.12: CDF of MTD for the different designs.

Fig. 3.11 shows differential power analysis traces from the first 4 bytes of a key in the 8-bit renaming design after 5k encryptions, and in the 8-bit randomized design after 30k encryptions. The differential power trace under the correct key guess (green) has peaks of higher magnitude than all incorrect key guesses (red) demonstrating a successful attack. Despite the larger number of encryptions performed to generate the traces, the peaks in the randomized design are less prominent.

Fig. 3.12 shows the cumulative distribution of MTD for key bytes in the three different designs. The DPA attack is repeated with different keys to extract a total of 64 key bytes. As would be expected, the plot shows that the key bytes can be extracted

most easily from the 8-bit renaming design. The 128-bit design is somewhat harder to attack due to the increased noise from parallel computations. The 8-bit design with randomization is found to be least susceptible to the side channel attack. The mean MTD values for the 128-bit, 8-bit renaming design, and proposed 8-bit randomized design are 9746, 1492, and 15983 respectively. This finding from simulation suggests that randomization helps to diminish the side channel susceptibility. In particular, the randomized 8-bit design is not more vulnerable than the 128-bit design, and requires an order of magnitude more encryptions to break than an ordinary 8-bit design. The randomized design is fabricated and its side channel resistance is evaluated in Ch. 4.

3.4 Summary

In this chapter, we presented a microarchitectural technique to improve energy efficiency of 8-bit implementation of AES. Our improved clocking methodology greatly reduces activity of data and key registers to a single update per round, which is at least 16x smaller than conventional 8-bit implementations. Register renaming eliminates the need for additional state registers to store intermediate results and minimizes data movement through registers, thereby saving energy. In comparison to the most efficient conventional 8-bit implementations, we consume 47% lower energy for an encryption operation with a 3x reduction in clock energy, while paying a 18% area cost. Our methodology can be extended to other sub-round implementations of AES like 32-bit. For a 32-bit implementation, energy inefficiencies in data movement and clocking are smaller and so would be the energy savings using our scheme.

Further, we have proposed a novel microarchitectural randomization scheme for serialized AES implementations to reduce their susceptibility to side channel attacks. We build on the register renaming architecture to create a design that can randomize the order of the sub-round operations while preserving correctness of the AES algorithm, and avoiding data hazards. In particular, the ability to randomize the

sub-round operations is enabled by a modified Enable Generation circuit and a permutation stage after MixColumns, and by rescheduling the key expansion and key addition across round boundaries. For an overhead of 36% area and 25% energy, our proposed architecture improves side channel resistance of the renaming design by an order of magnitude and removes the inherent vulnerability of 8-bit architectures relative to 128-bit designs. Our technique is well suited for low power/area applications and is an architecture that is suitable for implementing circuit-level countermeasures as well. Benefits of microarchitectural randomization can go beyond side channel resilience and also protect a design against targeted runtime attacks that would require an attacker to inject a fault during a specific computation. We fabricated our AES designs in a commercial 16nm FinFET technology and evaluate our testchip as described in the next chapter.

CHAPTER 4

TESTCHIP

In this chapter we describe the design methodology used to tape out our AES designs in a commercial 16nm FinFET technology. The chip was taped out through MOSIS in May 2018 and received back in December 2018. We also discuss our testing methodology and measurement results from the chip. In compliance with our NDA, we exclude reporting confidential information about the process technology in this thesis.

4.1 Design methodology

Our design flow comprises of RTL design, synthesis to obtain a gate level design, and physical design to generate layout. We further design a Printed Circuit Board (PCB) to test the fabricated chips.

4.1.1 RTL design and synthesis

A block level view of our system is shown in Fig. 4.1. As mentioned in Sec. 3.3, we implement four AES architectures: (1) an ordinary round-based 128-bit design, (2) our efficient 8-bit renaming architecture [32], (3) our 8-bit design with randomization [31], and (4) state-of-the-art reference 8-bit design [116]. The control block enables communication with all four AES design on the chip. It consists of a Finite State Machine (FSM) that takes as input clock, reset signals, a byte of plaintext and a byte of key per clock cycle from chip’s primary inputs. The control block uses FIFOs to interface with the different AES designs by feeding plaintext data of appropriate

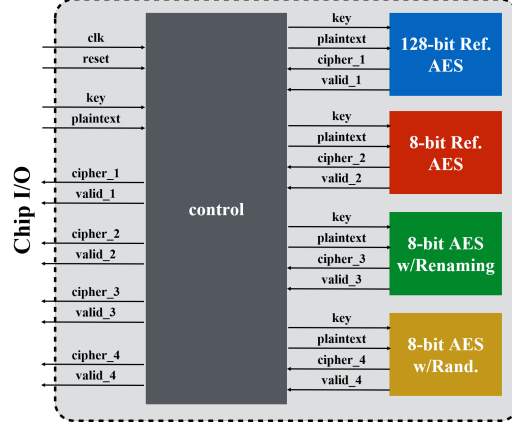


Figure 4.1: Block level view of designs implemented on testchip.

bit width and receiving ciphertext data. The ciphertexts are then serialized as 1-bit signals and output along with a valid bit to the chip’s primary outputs. Details of the AES architectures are presented in Ch. 3.

RTL design is the first step of implementation. RTL for all four AES designs are written by us and validated against an online tool. Synthesis is the process of converting RTL specification into an optimized technology-dependent gate-level design. Synthesis tools can optimize for area, speed and power subject to constraints. We use Synopsys Design Compiler (DC) for synthesis with a commercial 16nm FinFET standard cell library for technology mapping. Given that our goal is to come up with efficient lightweight designs, we optimize our designs for area subject to a timing constraint of 200 MHz clock frequency. Synopsys Prime Time (PT) is used to perform initial timing checks on the synthesized gate level netlist. PT performs static timing analysis and exhaustively checks the design for timing violations. The targeted clock frequency of 200MHz exceeds the I/O pad speed limitation, and we supply the clock from off chip through I/O, so we cannot operate above 200MHz even if the design is capable. Setup and hold timing are comfortably met by our designs at the foundry-specified worst case corners.

4.1.2 Physical design

Physical design is the process of creating an optimized layout for fabrication from the gate level design. It comprises several steps: design planning, power planning, placement and optimization, clock tree synthesis, routing and post-route optimization, chip finishing steps, and Design Rule Checks (DRCs). We use Synopsys IC Compiler (ICC) for physical design. Fig. 4.2 shows snapshots from different stages of the physical design flow for the 8-bit randomized AES design.

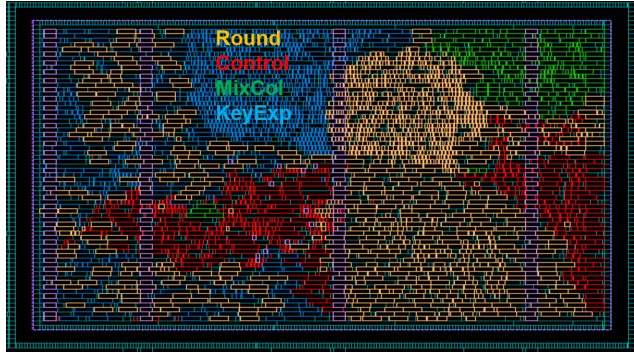
In design planning, a floorplan of the layout is created. It defines cell boundary and core area, and creates site rows for standard cell placement. I/O pads are placed outside the core area. We create a rectangular floorplan with a 50% target utilization for each of the AES designs. Due to schedule constraints we did not pursue optimization of higher utilizations but fairness in comparing results is assured by using the same utilization specification for all four AES architectures. Higher utilizations (such as 70%) would further reduce footprint of each design.

Power planning ensures a design with good power integrity, and a reduction in IR drop and electromigration. A power distribution network is created per vendor specification. A coarse-grained grid in the top two metal layers addresses IR drop while a fine-grained grid in the two lower metal layers addresses the di/dt requirements of the design. The spacing between power straps in each metal layer is determined based on current density limits of the metal layers, the design's power requirements, and routability. A snapshot of the created floorplan with power grid is shown in Fig. 4.2a.

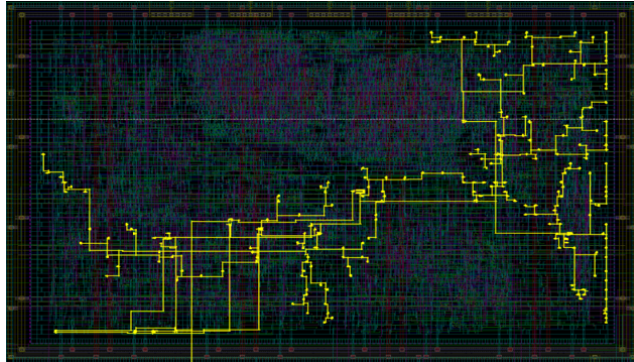
Following power grid creation, cells from the post-synthesis gate level netlist are placed in the various site rows of the floorplan core area. Legal placement of standard cells is obtained through an iterative process while optimizing for minimum area subject to timing constraints. Special physical only cells such as well taps to prevent latchup current and endcaps to meet DRC requirements at layout boundaries are



(a) Floorplan showing power grid



(b) Placed design



(c) Clock tree



(d) Routed design

Figure 4.2: Different stages of physical design of 8-bit Randomized AES.

placed first, followed by regular logic cell placement. Placement of cells for the 8-bit Randomized AES design is shown in Fig. 4.2b.

After placement, clock tree synthesis is performed to build the clock network that begins at the clock source and ends at the sequential elements of the design. The clock tree is optimized by buffer sizing/relocation to minimize clock skew between different fanouts and also reduce clock insertion delay. Clock nets are then pre-routed before proceeding to signal routing. The clock tree of the 8-bit Randomized AES design containing 579 sinks is shown in Fig. 4.2c.

As part of routing and postroute optimization ICC performs global routing, track assignment followed by detailed routing to determine the actual course of wires that connect the placed cells. Setup and hold time for critical process corners are checked in this step for timing closure while optimizing routes. Initial DRC checks are also performed to ensure routing abides by design rules. Setup timing is easy to meet due to the modest maximum frequency of the design which is limited by I/O pads. Following the initial routing step, we perform two incremental post route optimization steps to individually address hold time and DRC violations. Finally, we use a focal optimization step that targets only a single violation type to fix any remaining hold timing violations. Fig. 4.2d shows an example of a fully routed design. After timing closure of the fully routed design in ICC, we perform standard cell filler insertion to ensure well continuity. Sign-off timing checks using Synopsys PrimeTime passed while sign-off DRC checks using Synopsys IC Validator (ICV) had over 100 violations. The DRC violations were related to routing issues with I/O pins that were not on the routing tracks. Fig. 4.4 shows some examples of the DRCs that were manually fixed.

A snapshot of the entire system's layout containing all four AES designs and I/O pad placement is shown in Fig. 4.3. The layout area is pad limited as seen in the figure. Each AES design has its own isolated power network to enable accurate power measurements. The control logic from Fig. 4.1 is placed outside the four AES

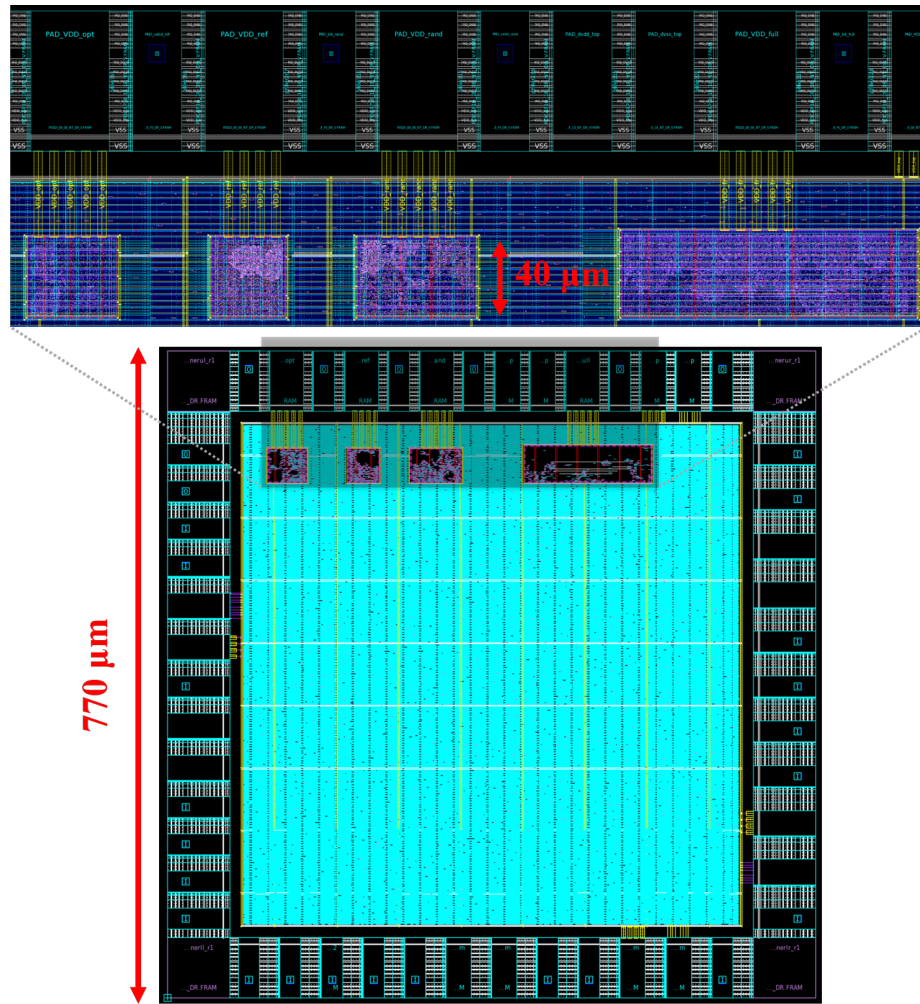


Figure 4.3: Layout showing all four AES designs.

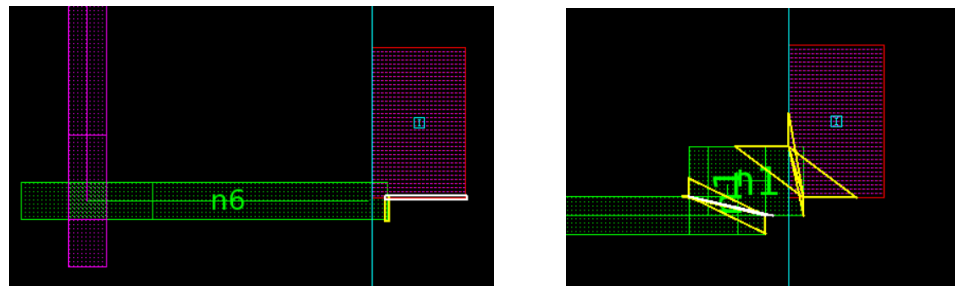
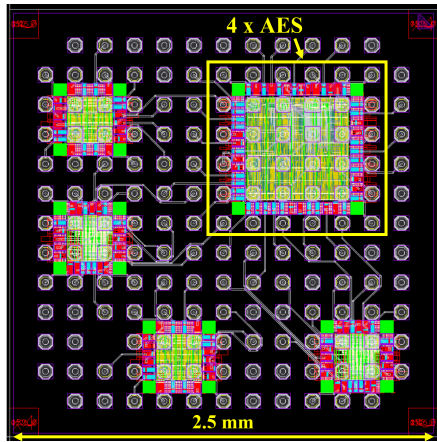
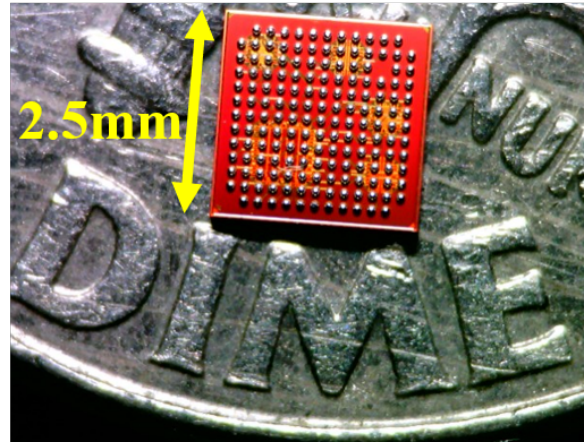


Figure 4.4: Examples of sign-off DRC violations on finished design that were manually fixed.



(a) Full chip layout.



(b) Unpackaged die on a dime.

Figure 4.5: Images of full chip with other designs from colleagues on same die. All designs have isolated power domains.

designs and has its own power network. The I/O pads are powered through a 1.8V supply while the AES designs and control logic in the core area are powered through individual 0.8V power supplies.

For full chip integration, designs from colleagues were also instantiated in the same die. The I/O pads from the different designs are routed to Re-Distribution Layer (RDL) bumps that are laid out in a 13x13 matrix over the entire die area. Synopsys ICC was used to perform RDL routing. FEOL and BEOL fills for manufacturability, and final DRC checks on full chip are performed using Synopsys ICV. Fig. 4.5 shows a picture of an unpackaged die containing the AES system among others. RDL routes are also visible in the image.

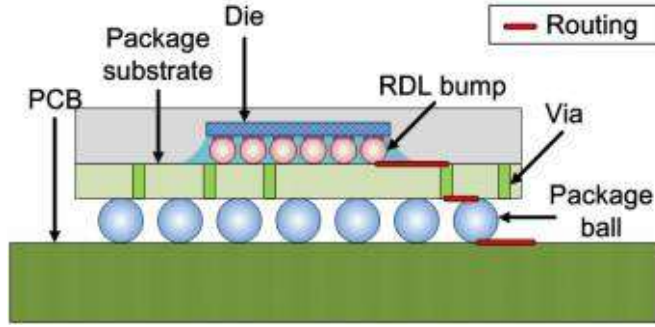
4.1.3 Chip packaging and Printed Circuit Board design

Flip Chip Ball Grid Array (FC-BGA) packaging is used to house the 2.5mm x 2.5mm fabricated die. Solder balls are laid out in a 13x13 grid at a 1mm pitch. Fig. 4.6a shows how RDL bumps of die are connected to solder balls in the package. Snapshots of the packaged chip are shown in Fig. 4.6b. The 168 package pins comprise:

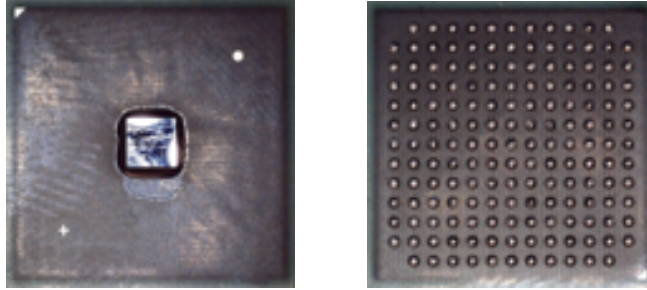
84 signal, 48 VDD and 36 GND pins. The four AES designs and control logic use 30 signal, 12 VDD and 8 GND pins. To test the packaged chips, we design a 4-layer Printed Circuit Board (PCB) using Autodesk Eagle. The boardview of the PCB is shown in Fig. 4.7a. The PCB integrates a BGA socket for testchip, an FPGA module with testbench, level shifters, screw terminals for power supply connections, and testpoints for side channel and energy measurements. The BGA socket (Fig. 4.7b) allows us to test multiple chip instances using the same PCB instead of soldering the chip directly to the board. The bottom part of the socket is mounted on the PCB using bolts such that the Surface Mount Technology (SMT) pads of the PCB are contacted by an elastomer guide of the socket. The packaged chip is placed in the socket such that its solder balls make contact with the socket’s elastomer guide which connects them electrically to the SMT pads of PCB. The top lid of the socket is secured using a torque driver. The socket manufacturer estimates that packaged chips can be swapped out 1000 times before the elastomer layer wears out and must be replaced.

4.2 Chip Testing

In this section we present results on efficiency and side channel resilience of the four AES designs based on measurements from the testchip. The PCB designed for chip testing (Fig. 4.7b) houses an Artix 7 FPGA (Cmod A7 [34]) and a Flip Chip Ball Grid Array (FCBGA) socket that holds the chip. The FPGA handles all communication with the testchip. A MATLAB program sends plaintext data to the testchip via FPGA and receives ciphertext data from the testchip in a similar fashion. The received ciphertext is validated in MATLAB against a software implementation of AES to continually check correctness of chip output. The FPGA operates at 3.3V while the testchip uses an 0.8V core voltage and 1.8V I/O voltage. Level shifters



(a) Illustration of flip-chip packaging [110].



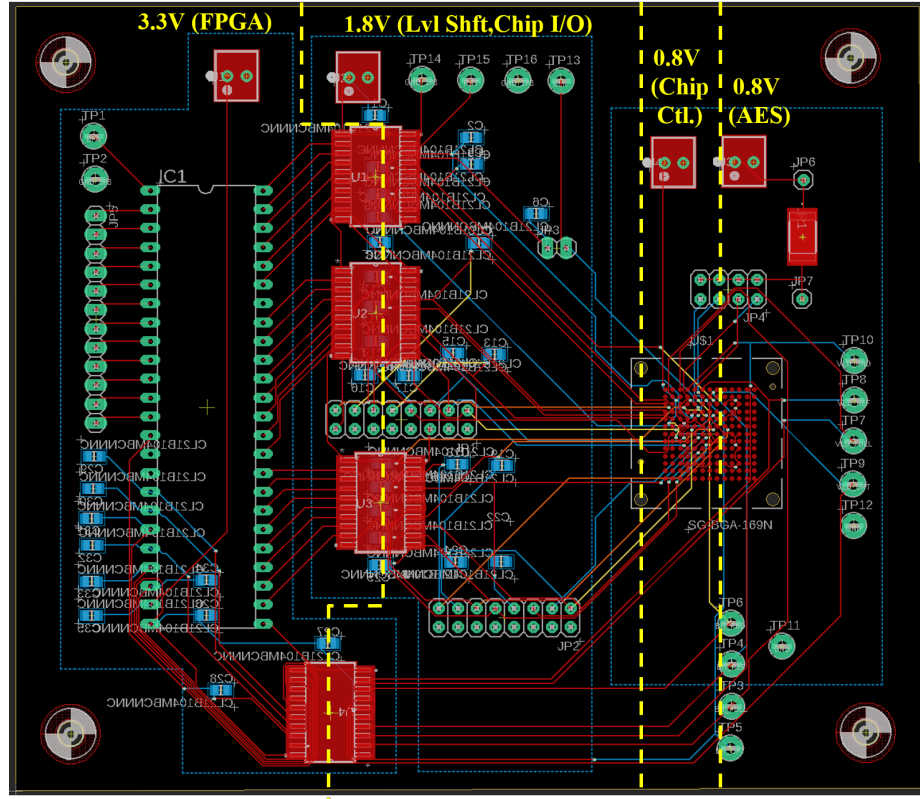
(b) Images of packaged chip showing front and back side.

Figure 4.6: Chip packaged in a Flip Chip Ball Grid Array

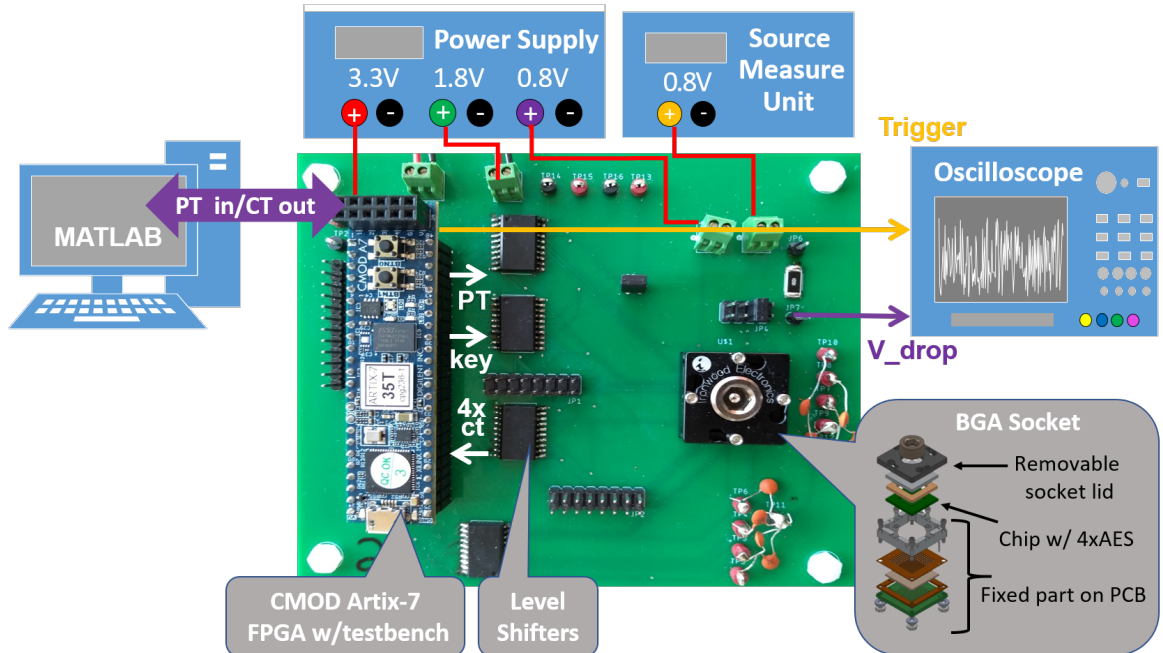
translate voltage levels appropriately between FPGA and testchip I/O. A Keysight Source Measure Unit B2901A [60] is used to supply power to individual AES modules and also measure current with 100fA precision. I mentored Samuel Allen in the summer of 2019 and would like to acknowledge his help with some aspects of test setup and data collection.

4.2.1 Efficiency, power

Power measurements are made using the Source Measure Unit (Fig. 4.7b) by powering each AES design in isolation. Power consumption per encryption at different clock frequencies is shown in Fig. 4.8a and the slope of the line plot ($\mu W/MHz$) is determined for each design. The renaming architecture has a power consumption of $0.42\mu W/MHz$ which is 13x lower than a 128-bit design which consumes $5.75\mu W/MHz$. Power does not equate to efficiency because the designs have different throughput. The metric of $Mbps/\mu W$ describes efficiency of a design. It remains

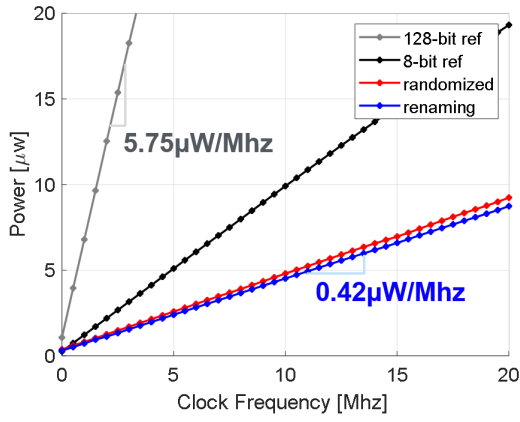


(a) PCB boardview showing isolated power domains for different components

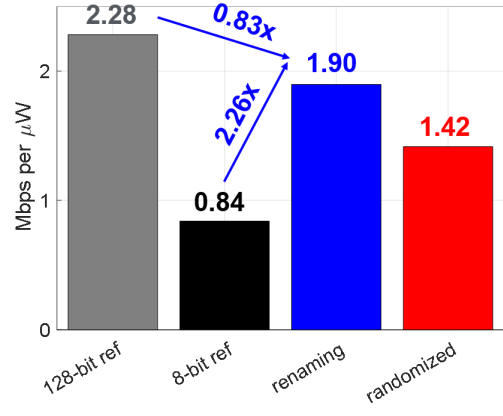


(b) Data collection setup to perform automated measurement from testchip. BGA socket [53] houses the chip.

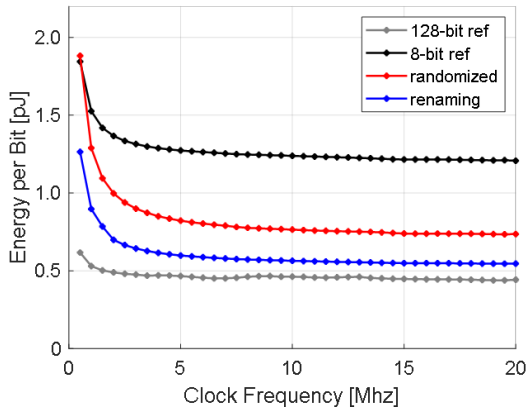
Figure 4.7: AES chip test setup.



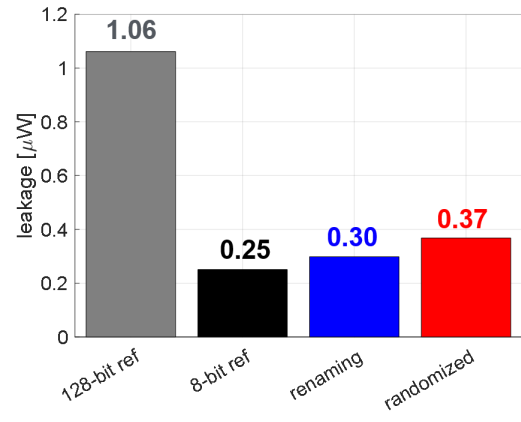
(a) Power (μW) Vs Clock frequency



(b) Efficiency (Mbps/ μW)



(c) Energy-per-bit (pJ) Vs Clock frequency



(d) Leakage Power (μW)

Figure 4.8: Power and energy comparison of AES designs based on testchip measurements.

Table 4.1: Performance comparison of AES designs with testchip measurements obtained at 20MHz clock.

	128-bit Ref.	Renaming	Randomized	8-bit Ref.
Energy (pJ/bit)	0.44	0.55	0.74	1.20
Power (μW)	113.28	8.74	9.24	19.32
Throughput (Mbps)	256.00	16.00	12.54	16.00
Cell Area (μm^2)	3066	886	1201	750
Die Area (μm^2)	6450	1800	2400	1480
Latency (cycles)	10	160	204	160

constant as clock frequency is adjusted to increase throughput for proportionally more dynamic power. In terms of efficiency from Fig. 4.8b, the 128-bit datapath is the most efficient at $2.28 \text{ Mbps}/\mu W$ due its to 16x higher throughput than the 8-bit designs. The renaming architecture is competitive at $1.90 \text{ Mbps}/\mu W$ and is twice as efficient as the state-of-the-art reference 8-bit design ($0.84 \text{ Mbps}/\mu W$). Even our Randomized architecture is more efficient at $1.42 \text{ Mbps}/\mu W$. Efficiency can also be measured in terms of Energy-per-bit and similar trends are seen in Fig. 4.8c. Both the renaming and randomized designs have sub-pJ/bit energy efficiencies as tabulated in Tab. 4.1. Leakage power, measured by turning off the clock, is correlated to design area as seen in Fig. 4.8d. The 128-bit design exhibits highest leakage power because it is about 4x larger in area than the 8-bit designs. Cell area, latency and throughput of the four designs are tabulated in Tab. 4.1.

The efficiency of all designs can further be improved with voltage scaling and we reassured this experimentally for the Renaming architecture. Though we did not design for low voltage operation the testchip was functional down to 0.6V resulting in improvements of 2x in efficiency and 33% in leakage power (Fig. 4.9).

4.2.2 Side Channel resilience

Resilience of the 8-bit AES designs to side channel attacks is analyzed by performing a Hamming Distance based DPA attack during the final round of encryption. A

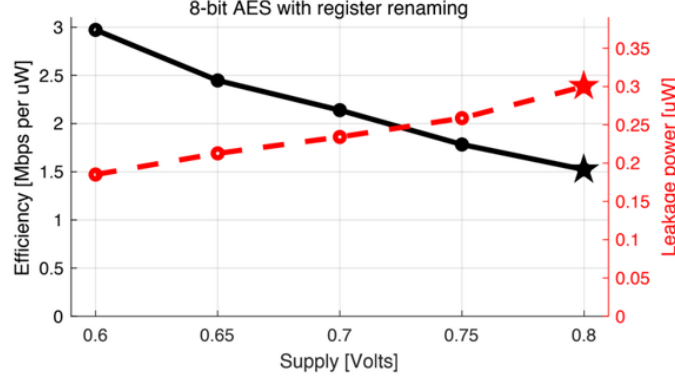
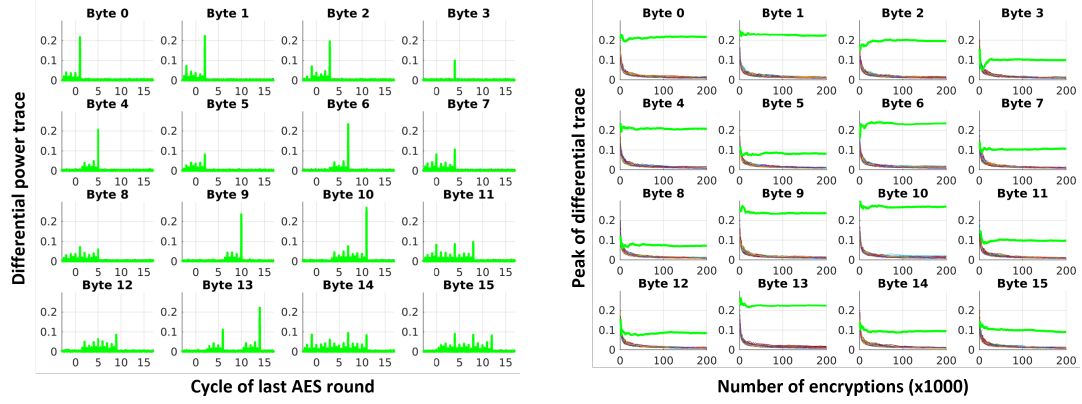


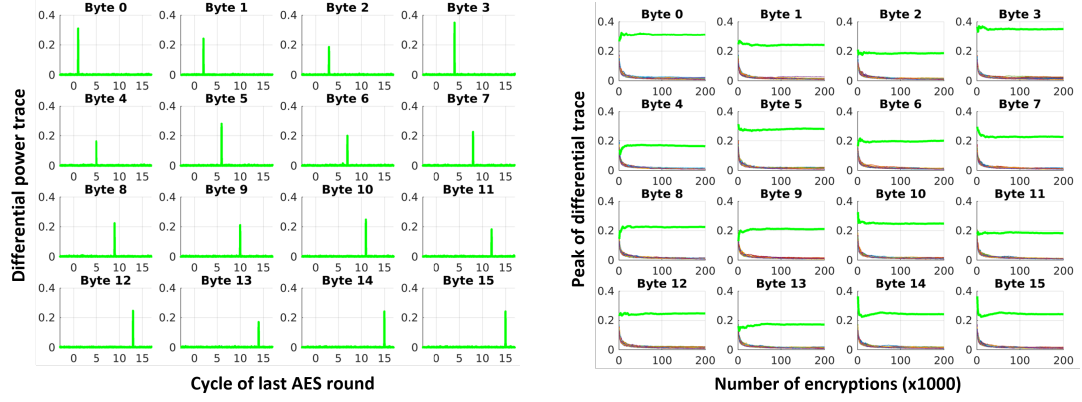
Figure 4.9: Effect of voltage scaling on efficiency.

background on DPA is given in Sec. 1.3.1. Power traces are captured by measuring the voltage drop across a 1k supply-side resistor as illustrated in Fig. 4.11a. With our measurement setup (Fig. 4.7b), we are able to capture 20K traces/hr using MATLAB with Keysight MSOX4154A (700Mhz probe) oscilloscope [61] and USB data transfer. Since bytes are processed one at a time in an 8-bit datapath, Hamming Distance between consecutive bytes at the S-box input is chosen as the selection function to partition the power traces (Fig. 4.11b). This selection function is meaningful because of the side channel leakage in power consumption when a data byte (green star in Fig. 4.11b) overwrites the previous one at S-box input.

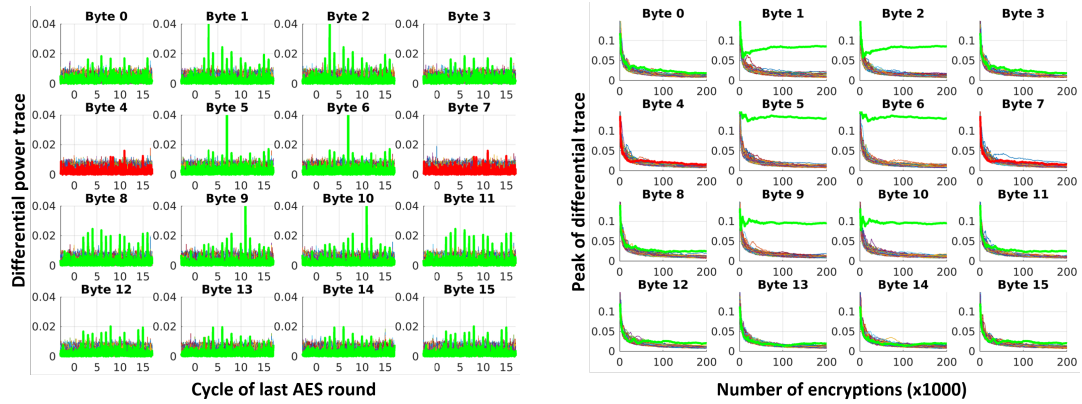
The attacker observes the ciphertext byte ct and guesses the key byte k to compute the value of byte b at S-box input. In the next clock cycle, the procedure is repeated with the new observed ciphertext byte to compute the S-box input byte b' that overwrites byte b , and a Hamming Distance HD between bytes b and b' is computed. In this manner, power traces are aggregated in $Class_1$ if $HD > 4$ and in $Class_0$ if $HD < 4$. The **differential trace** is computed as the average difference between traces in the two classes. The attacker computes a differential trace for each key guess for the byte he wishes to attack. A peak in the differential trace indicates a correlation between the key guess and power measurement. With enough measurements the differential trace of the correct key byte guess will consistently show



(a) 8-bit reference



(b) 8-bit renaming



(c) 8-bit randomized

Figure 4.10: Differential and DPA traces with Hamming distance leakage model.

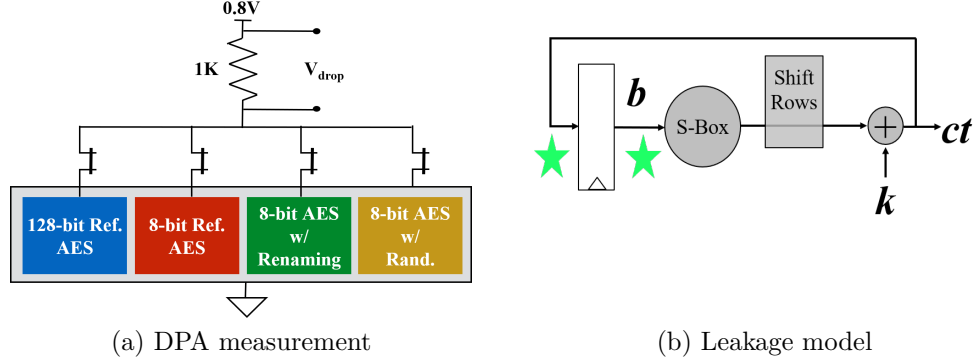


Figure 4.11: Differential Power Analysis

a higher peak than the differential traces of incorrect key byte guesses. We use the term **DPA trace** of a key byte guess to denote the maximum values of the differential traces when varying the number of encryptions used for the attack.

The differential and DPA traces at the end of 200K encryptions are shown in Fig. 4.10 for the 8-bit AES designs. All 16 key bytes are attacked and the location of the differential trace peak varies depending on the clock cycle of the last AES round in which a particular key byte is processed. In Fig. 4.10, a successful DPA attack is indicated by green lines for the differential and DPA traces of the correct key; whereas a failed attack with 200K encryptions is indicated by corresponding red lines.

- In the renaming design key bytes are processed sequentially one per clock cycle and the peaks in differential trace also follow suit (Fig. 4.10b).
- In the randomized architecture key bytes are processed in a random order and a particular key has equal probability to be computed in any of the 16 clock cycles of the last AES round. This results in 16 smaller peaks in the differential trace (Fig. 4.10c).
- The reference 8-bit design also has multiple peaks in the differential trace (Fig. 4.10a) but for a different reason. In this architecture, data bytes pass

Table 4.2: Measurements to Disclosure using Hamming Distance DPA on the 8-bit AES designs

	Renaming	Randomized	8-bit Ref.
Mean MTD	761	118,400	1897

through register-to-register in the shift rows pipeline several times before appearing at the S-box input.

The resilience to DPA is quantified in terms of Measurements to Disclosure (MTD) [107] - the number of measurements required to distinguish the correct key guess from incorrect ones. MTD is defined as the cross-over point between the DPA trace of the correct key byte guess and the maximum of DPA traces of all the wrong key byte guesses. The DPA attack is repeated 40 times with all 16 key bytes on the 8-bit AES designs and the average MTD values are listed in Tab. 4.2. The renaming design is easy to break with just 761 encryptions required on average for a successful DPA attack. The efficiency of the renaming design in eliminating unwanted switching results in a lower background noise power and makes the design more vulnerable to DPA. The 8-bit reference design in comparison has slightly higher MTD, apparently due to increased noise power. The randomized design is much harder to break ($\text{MTD} > 100,000$ encryptions) as the signal is spread out in time over 16 clock cycles as was shown in Fig. 4.10c. Adding randomization capability to the renaming design increases MTD by two orders of magnitude. While it is hard to draw absolute conclusions, the significant increase in MTD indicates that randomization can help slow down the attacker from retrieving the secret key.

4.3 Summary

In this chapter we presented the design methodology for taping out our AES architectures in a commercial 16nm FinFET technology. Our automated testing methodology enables functional verification, energy measurements and side channel

analysis of the AES designs. In line with our simulations, the renaming architecture exhibits 2x efficiency improvements over the state-of-the-art 8-bit reference design and is the most efficient 8-bit AES to date to the best of our knowledge. The randomized design shows promise to improve side channel resilience which is an important concern with 8-bit datapaths of AES. Both of our AES designs consume sub-pJ/bit energy making them attractive candidates for low power applications in resource constrained scenarios. With techniques such as voltage scaling, further improvements in efficiency can be achieved.

CHAPTER 5

PACKAGE IDENTIFICATION

In this chapter we propose and evaluate COUNTERFOIL, a system that uses inexpensive cameras to check intrinsic variations in semiconductor packaging as means of verifying IC provenance. We name our system COUNTERFOIL both to reflect its aim of foiling counterfeits, and because the enrollment records it uses are analogs for counterfoils kept by issuers of cheques¹.

IC manufacturing often involves off-shore foundries for fabrication, a packaging house where the bare die is mounted within an encapsulating package, and system integration where packaged chips are soldered onto a PCB. The supply chain for packaged ICs can involve several distributors before the IC gets installed in a system. Securing the supply and distribution chain of ICs is important to prevent counterfeit parts. Besides monetary risks of billions of dollars counterfeit ICs can pose serious security threats to critical systems in defense and healthcare as discussed in Sec. 1.4. Some examples of counterfeit ICs are unauthorized copies (overproduction), remarked old parts (recycling) and ICs with misrepresented speed grades.

Existing strategies for preventing counterfeits parts from being used in systems can be broadly classified as either trying to detect anomalies, or else authenticating individual chip instances that are trusted. A common approach in counterfeit identification is to apply a battery of tests to a part in order to evaluate whether it

¹Oxford Dictionary defines counterfoil as “The part of a cheque, receipt, ticket, or other document that is torn off and kept as a record by the person issuing it.” <https://en.oxforddictionaries.com/definition/counterfoil>

is consistent with the expectations based on known good parts. The applied tests include physical inspection (visual [8], x-ray imaging, microblast analysis of the surface, spectroscopy, ion chromatography), electrical inspections [65, 16], and checking for aging using silicon odometers [5], ring oscillators [46], dynamic current signatures in adders [117], or other circuits that change in a measurable way with use. If any tests reveal an anomaly, the part can be deemed counterfeit. Anomaly detection techniques are used as part of qualification procedures by the US Department of Defense to minimize the risk of counterfeits, but “may not definitively distinguish authentic parts from counterfeit parts” [97]. Machine learning and neural network based techniques [99] detect anomalies in microscopic features to classify genuine and counterfeit parts. Unlike these approaches our technique relies on extracting unique fingerprints from individual parts to authenticate provenance and thereby prevent use of counterfeits.

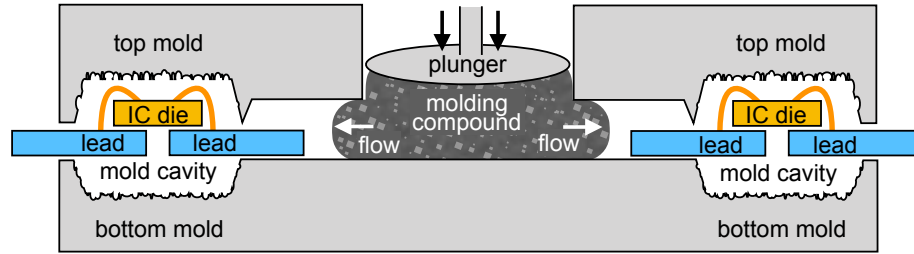
An alternative to anomaly detection is to identify and authenticate individual part instances using unique or hard to clone features. If a part is trusted at one point in time, and later a part can be validated as being the same one that was earlier trusted, then a judgment can be made that the part is still trustworthy. Well-known non-microchip versions of this style of object authentication include human fingerprints [39] and anti-counterfeiting features in currency [85]. Similarly, Physical Unclonable Functions (PUFs) are a type of physical fingerprint that can be used for authentication of parts. PUFs can be based on random delays in silicon [41], power-up fingerprints of Static Random Access Memory [44, 49, 102], randomly scattered dielectric particles in a protective coating [112], or unique Radio Frequency emissions [28, 26], among many others.

Several existing strategies for validating provenance of microchips are implicitly relying on the IC package as the basis for trusting the enclosed silicon die. The DARPA SHIELD project aims to embed inside IC packaging a secure dielet that

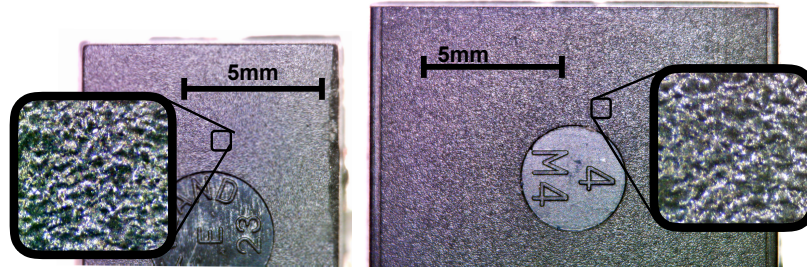
can be interrogated wirelessly with a smart-phone like device to validate provenance of the part [67]. A company called Applied DNA Sciences offers a botanical DNA taggant that can be applied to various goods including microchip packages [48] to support traceability through distribution. To date, working with the Defense Logistics Agency (DLA) of the US Department of Defense, the technology has marked over 700,000 microchips [79]. Both package-embedded dielets and package tagging have an underlying assumption that an adversary cannot easily swap a microchip out of its package, and therefore validating the package provenance suffices to validate the provenance of enclosed microchip.

5.1 Transfer Molding for IC packaging

Transfer molding (Fig. 5.1) is the typical procedure for packaging of high-volume integrated circuits [15, 25]. Most DIP (Dual In-line Package), SMT (Surface-Mount Technology), and QFP (Quad Flat Package) packages are created this way, as well as more advanced packaging styles such as system-in-package. In the transfer molding process, each silicon die is first attached to a metal leadframe, and the pads from the die are wire-bonded to the individual leads to create electrical connections. Each leadframe-mounted die is then placed in a mold cavity, with the leads extending out the side of the cavity. A plunger liquefies pucks of epoxy molding compound using temperature and/or pressure. The liquefied compound flows through runner channels into the mold cavity to surround the die and form the shape of the package. After the compound solidifies, the molds are released, and the leads are separated from the remainder of leadframe, which is discarded. The metal leads protruding from the formed package are now the pins of the packaged chip that will connect it to a printed circuit board. Further details on the many packaging styles for integrated circuits can be found in a popular textbook on the topic [111].



(a) Transfer molding of package for IC on leadframe



(b) Surface texture of molded packages

Figure 5.1: Transfer molding is the mechanism used for packaging most high-volume microchips.

Several sources of variability in transfer molding can impart unique features to a package surface. The mold has a surface roughness that gets imprinted onto the package. The surface texture of the mold changes over time as residue material accumulates on the mold, and molds require cleaning to mitigate this build up [50]. Additionally, the molding compound itself, and its curing, contribute a certain amount of unpredictability. The molding compound is an epoxy that contains a number of fillers including crushed quartz or alumina that comprise 75% or more of the compound, and provide thermal conductivity. The size of the filler particles can range from $20\text{-}100\mu\text{m}$, and the orientation and distribution of filler particles in the package is unpredictable. The package during post-mold curing also experiences shrinkage, cracks, porosity, and voids [109]. Due to aforementioned variation sources, even chips packaged in the same mold could have differences in their package surface.

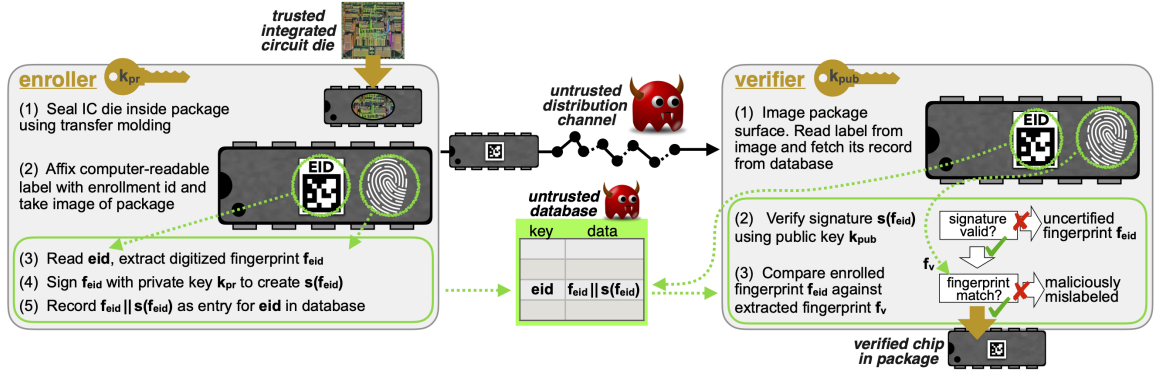


Figure 5.2: Protocol for package fingerprinting. Trusted enroller labels each package and then enrolls it by extracting and then signing a set of keypoints associated with the package. Verifier compares the enrolled keypoints against the package to determine whether the package is consistent with its label.

Algorithm 1: ENROLLCHIP

Input: Image img of chip surface with marker attached. Private key k_{pr} for signing messages.

- 1 $eid \leftarrow readMarker(img)$
- 2 $f_{eid} \leftarrow extractKeypoints(img, r, \theta, w_{enroll})$
- 3 $s(f_{eid}) \leftarrow Sign(k_{pr}, f_{eid})$
- 4 $database[eid] \leftarrow f_{eid} || s(f_{eid})$
- 5 **return**

Algorithm 2: VERIFYCHIP

Input: Image img of chip surface with marker attached. Public key k_{pub} to check signatures.

Output: Success or failure to verify chip as authentic according to the identity on its label

- 1 $id \leftarrow readMarker(img)$
- 2 $f_{eid} || s(f_{eid}) \leftarrow database[id]$
- 3 **if** $VerifySignature(k_{pub}, s(f_{eid}))$ **then**
- 4 $f_v \leftarrow extractKeypoints(img, r, \theta, w_{verify})$
- 5 **if** $score(f_{eid}, f_v) > threshold$ **then**
- 6 **return** *success*
- 7 **return** *fail*

5.2 CounterFoil anti-counterfeiting scheme

COUNTERFOIL uses package surface features to authenticate provenance of individual chips as shown in Fig. 5.2. The two participants in the scheme that interact with the chip are denoted as the enroller and a verifier. The enroller acts on behalf of a chip manufacturer that wishes to sell parts with an assurance of provenance. The verifier is a customer that has purchased the chips on the market and wants to check whether they are legitimate. Both the manufacturer as enroller, and customer as verifier, have incentives for participating in the presented scheme. The chip manufacturer can make their products more attractive by offering an assurance that authentic parts bearing their branding can be verified as produced by them. Importantly, they can accomplish this without needing to trust every point in their distribution channels. The chip customer is incentivized to participate because systems that are free from counterfeit chips can avoid costly failures or recalls that are caused by counterfeits [94].

The enroller extracts fingerprints from package surface features using image processing and publishes information about enrolled chips to a public database. Integrity of database entries is assured by digital signatures. The enroller holds a private key k_{pr} for signing messages, and gives the corresponding public key k_{pub} to any parties that wish to act as verifiers. Our implementation uses the simplifying assumption of pre-existing public keys for enroller and verifier, but in practice this could, for example, rely on a trusted certificate authority. The enroller uses the private key to sign database entries when writing them, and the verifier uses the enroller’s corresponding public key to check the signatures when reading from the database. More details about the enrollment (Alg. 1) and verification (Alg. 2) procedures are given below. Details of the image processing performed in enrollment and verification are deferred to Sec. 5.3.

5.2.1 Enrollment

The enrollment procedure should occur as part of the packaging of an IC. The IC should be trusted at the time of packaging, as the goal is to later tie provenance back to this point. Each die is sealed inside of a molded plastic package as usual by means of transfer molding (see Sec. 5.1 and Fig. 5.1). After the package hardens and cures, a label with a computer-readable identification marker is affixed to the surface of the package. The marker represents an insecure numerical identifier of the chip instance, similar to a serial number, which we denote as its *eid* (enrollment identifier). The enroller then takes an image that captures both the marker, and the package surface in the vicinity of the marker, from which the fingerprint will be extracted. A digitized enrollment fingerprint f_{eid} is extracted from the image, using a procedure that will be explained in Sec. 5.3.2. The date of manufacture and other metadata can be appended to the fingerprint at this point. The enroller creates signature $s(f_{eid})$ by digitally signing fingerprint f_{eid} using private key k_{pr} (Alg. 1, line 3). An entry is added to the public database to associate the identifier *eid* with $f_{eid}||s(f_{eid})$ (Alg. 1, line 4). Once the chip is enrolled to the database, it is released into distribution channels.

5.2.2 Verification

The verification procedure checks authenticity of chips at the end of distribution. The verifier takes an image of the chip that includes both the marker and the package surface in the vicinity of the marker. The insecure identifier (*eid*) of the marker is extracted from the image. The enrolled data $f_{eid}||s(f_{eid})$ for this identifier is accessed from the database (Alg. 2, line 2). The validity of signature $s(f_{eid})$ is checked using the public key k_{pub} of the enroller (Alg. 2, line 3). The enrolled fingerprint f_{eid} is compared against a new fingerprint f_v that is extracted from the relevant area of the chip package surface. If the similarity score exceeds a chosen threshold, then the

package surface is determined to match the record (Alg. 2, line 5). The chip is verified as authentic only if the digital signature is valid, and the fingerprints match. The validity of the signature ensures that the enrolled fingerprint in the public database was created by the enroller and has not been modified. The fingerprint match ensures that the enrolled data is not being used to authenticate a chip other than the one that was enrolled, a scenario that would arise if a label was copied or transferred from one chip to another. The verification procedure is currently performed on a workbench in our lab, but could later, for example, be integrated into a pick-and-place machine at the end of distribution that picks chips from reels and places them appropriately onto printed circuit boards.

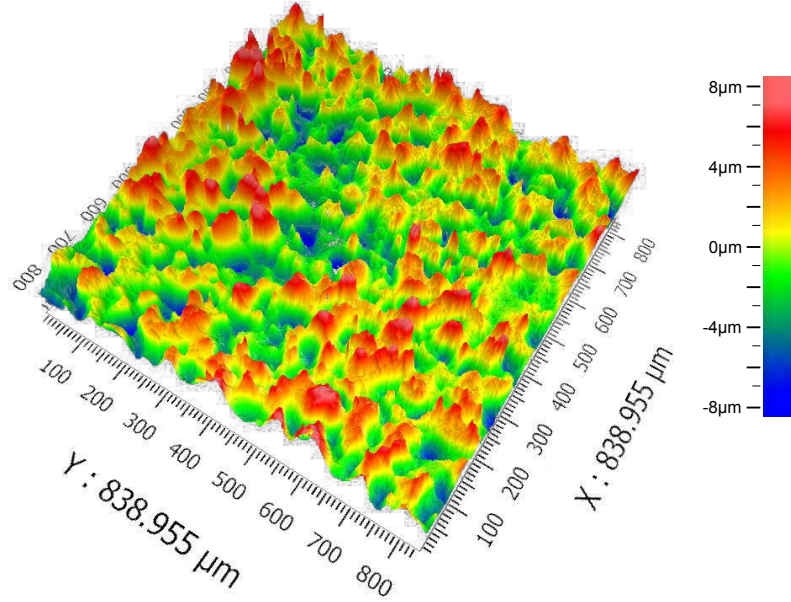
5.2.3 Attacker Capabilities and Security Considerations

The attacker considered in this work is a profit-motivated counterfeiter that forges chips for purpose of selling them on the market. This type of profit-seeking attacker is responsible for prior counterfeit parts found in sensitive systems, but note that it does not include nation-state attackers that may spend large amounts of money to create malicious forgeries to bring down targeted high-value systems. For a profit-seeking attacker, if the effort of forging chips exceeds the selling price of the chip on the market, there is no incentive to forge the chips. At the same time, the cost for anti-counterfeiting technology in commodity parts cannot exceed what the producer or consumer of the parts is willing to spend for the guarantee of provenance.

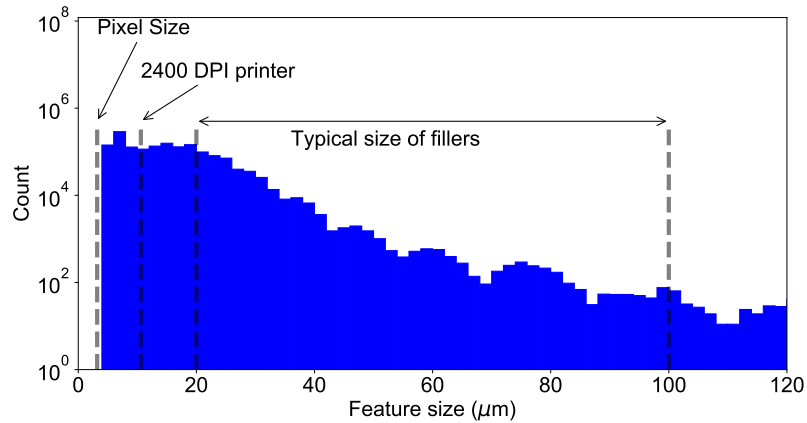
The security of our approach relies on assumptions similar to those in earlier work on certificates of authenticity [28]. Our assumptions relate to the enrollment and verification protocol, the uniqueness of package fingerprints, and the difficulty of creating forged chip packages that match legitimately enrolled fingerprints. Among these three, the first is intended to be uncontroversial, and the latter two are supported by experimental data in the paper.

1. **Protocol Integrity:** We make the standard assumption that an adversary is not able to obtain the enroller’s private key or forge digital signatures without having the private key. We assume that the enroller is trusted to only package legitimate integrated circuits, and to enroll only these packages with the private key k_{pr} .
2. **Unique Fingerprints:** We rely on the fact that package fingerprints created under ordinary conditions are unique and are identifiable via image processing. Specifically, an enrolled fingerprint from one package will not be deemed a match for any package other than the enrolled one. Fingerprint uniqueness binds the enrolled data to a specific chip instance. If labels are later affixed to chips other than the enrolled, the enrollment data associated to the label will not match the chip characteristic. This prevents an adversary from successfully copying or transferring labels across chips.
3. **Difficulty of Package Forgery:** We assume, and then support experimentally, that package fingerprints are random and difficult to control. This prevents an adversary from creating a new package surface that matches a legitimate enrolled fingerprint. We support this assumption by showing that even chips from the same mold have different fingerprints. This implies that even possession of an identical mold will not enable an adversary to successfully forge packages and therefore forgery requires a more advanced manufacturing process than what industry uses for packaging chips. Regardless of the process used to create forgeries, an adversary will have to create recognizable features with sizes on the order of $10\mu m$ (see Fig. 5.3). Besides attempting to clone the package surface an attacker could print a label with features from a legitimate chip. However, the printing task is seemingly out of reach of many technologies such as high-end 2400 DPI printers, which have a dot size of $10.6\mu m$ and can

only print reliable features at a much larger scale than its dot size. Aside from forgery, an adversary might transfer the package from a legitimate part to a counterfeit IC, but there would be no profit motive to this, as it would destroy a legitimate chip to create a single forged chip.



(a) Package surface profiled using Zygo Nexview [118]



(b) Extracted feature sizes from image processing.

Figure 5.3: Size of features extracted from images of package surfaces using OpenCV implementation of ORB algorithm as discussed in Sec. 5.3.2

Note that an adversary could make a chip unverifiable by simply removing or irreparably damaging its label. We view this as a reliability concern more than

a security concern, as counterfeiters would not earn profit by disabling the trust-enhancing technology from legitimate parts. Nonetheless, the paper labels we use in our prototype system would likely be replaced by a more robust marking when deploying COUNTERFOIL at production scale.

5.3 Image Processing and Analysis

Our system relies on image processing as part of enrollment and verification. Enrollment generates a digitized representation of recognizable features within a selected area of the package surface. Verification later scores the record of enrolled features against a new image of the package surface. In this section we describe the computer vision algorithms used. Our algorithms are written in C++ using OpenCV [21] for the image processing.

5.3.1 Aruco marker labels and detection of ROI

Our system uses computer-readable labels (Fig. 5.2) to represent the purported identity of a package. We also use the labels as fiducial marks to determine the Region Of Interest (ROI) in an image used for both enrollment and verification. It is to be noted that one could choose to use the entire image for enrollment/verification but this would increase runtime as discussed in Sec. 5.4. Aruco, the specific marker system that we use, is a square-based fiducial marker system with binary codes [40]. Aruco marker dictionaries are configurable, allowing for an arbitrary marker capacity (in bits) and number of markers. We use Aruco markers to label the chips with the search tag of the public database. The four corners of the marker allow for detection of image orientation (pose estimation) which we leverage to determine the ROI for further processing. Fig. 5.4 shows a detected marker with its top-left corner used to determine the center of ROI at a distance $\langle r, \theta \rangle$ relative to the marker. Depending on whether the image is being processed for enrollment or verification, the ROI selected

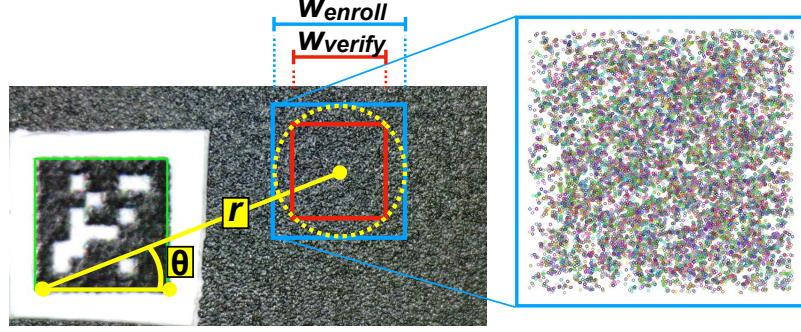


Figure 5.4: Image of chip with affixed marker. The position of enrollment ROI is shown by the blue box, and the callout shows the keypoints extracted from the ROI. The ROI that would be used for verification is the smaller red box. The size and position of both ROIs are defined relative to the marker, as shown by annotations in yellow.

from the image would be either ROI_{enroll} (blue square) and ROI_{verify} (red square). Both squares are centered at the same point, and have a size that is defined relative to the marker size for magnification invariance. The width of the larger square is $w_{enroll} = 2mm$, and the width of the smaller square is $w_{verify} = w_{enroll}/\sqrt{2}$. The difference in ROI sizes ensures that the ROI from enrollment will always contain the ROI from verification regardless of rotation. Consider the yellow circle in Fig. 5.4 which is centered at point $\langle r, \theta \rangle$. Regardless of the image orientation, the red square will always be contained within the circle, and the blue square will always contain the circle. Therefore, the blue square (ROI_{enroll}) will always contain the red square (ROI_{verify}). Further, ROI_{enroll} is chosen larger than ROI_{verify} to save runtime, as the verification involves more processing steps than enrollment. In our experiments we use $r = 5mm$ and $\theta = \pi/8$.

5.3.2 Feature Enrollment

The enrollment process extracts distinctive features from an image which are suitable for matching and object recognition, and stores them as compact feature descriptors. A number of well-known image processing techniques exist for feature detection

and description, such as Scale Invariant Feature Transform (SIFT) [72], Oriented FAST and Rotated BRIEF (ORB) [96], Binary Robust Invariant Scalable Keypoints (BRISK) [68], and Speeded-Up Robust Features (SURF) [12]. These techniques are commonly used in applications such as image stitching, where image alignment requires finding corresponding points of objects in two different images that contain the objects. Our work is agnostic to the choice of algorithm, but based on empirical evaluation (as will be discussed in Sec. 5.4.4) we choose ORB.

We first pre-process the image (ROI) using Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve the contrast and tolerance to variation in lighting intensity. We then use OpenCV’s implementation of ORB to extract image features. The keypoints are detected by Oriented FAST algorithm and described by 256-dimensional rotated BRIEF descriptors [96]. Similarity between two keypoints can be evaluated using *feature distance*, which is the Euclidean distance between two keypoints in the 256-dimensional feature space. The keypoints also have associated positions within an image, and we will use *pixel distance* to denote the Euclidean distance in two dimensions between pixels in an image. For the sake of predictable runtime, we restrict the number of keypoints to $1,000/mm^2$ of package surface. Fig. 5.4 shows the keypoints extracted from the region of interest.

The enrolled features are stored in a public database along with a digital signature (Fig. 5.2). The NIST Digital Signature Standard (DSS) establishes three algorithms for signatures, RSA, Digital Signature Algorithm (DSA) and Elliptic Curve DSA (ECDSA) [59]. We choose DSA in our implementation, but this can be replaced by either of the other algorithms with minimal performance impact. For hashing function, SHA-3 is chosen because it is the latest Cryptographic Hash Standard issued by NIST [36]. More specifically, the enrollment data is hashed using SHA3-256 and subsequently signed with the enroller’s private key using an implementation of DSA with 3072-bit private key as from the open-source Crypto++ library [3].

5.3.3 Feature Verification

Verification compares the enrolled keypoints against the ROI of a new image in order to compute a similarity score. The integrity of enrolled keypoints is first verified by checking the digital signature. When a new image is captured for verification, its ROI is identified relative to the marker, and keypoints are extracted from the ROI. This mirrors the corresponding steps performed in feature enrollment, so we don't repeat their description here. The processing performed with the verification keypoints is as follows.

5.3.3.1 Feature matching and RANSAC based homography computation

Two images of the same planar surface taken from different perspectives are related by a homography, which is a geometric model that maps feature positions in one image to the corresponding positions in the second image. Estimating the homography requires finding enrollment and verification keypoints that are similar and therefore likely to be representations of the same feature on the package surface. We find such points by performing nearest neighbor matching using Open CV FLANN (Fast Library for Approximate Nearest Neighbors) [82] matcher, and then evaluating quality of matches using a standard approach based on ratio of feature distances [72] as described here. For every keypoint k_i in ROI_{enroll} , we find its two closest (in feature distance) keypoints (k'_1 and k'_2) from ROI_{verify} and compute from their Euclidean distance in feature space a ratio score $r_i = \frac{\|k_i - k'_1\|_2}{\|k_i - k'_2\|_2}$. A low ratio indicates that keypoint k_i is significantly more similar to its best match k'_1 than to its second best match k'_2 , which implies that k_i and k'_1 are likely to be corresponding points in the two images [72]. The 50 keypoint pairs with the lowest ratios (i.e., the best matches) are used as the basis for estimating a homography with the RANSAC algorithm. Increasing the number of matches will reduce the chance of RANSAC reaching consensus

on an incorrect homography, but increases the expected number of random samples required to find consensus.

RANSAC (Random Sample Consensus) [38] is an algorithm to estimate a model from noisy data that contains both inliers and outliers. In our case, the computed model is the homography, and the data are the 50 selected keypoint pairs. RANSAC first samples four keypoint pairs from the set and calculates from them a homography matrix as in Eq. 5.1, where the 3x3 matrix is the homography, and P_e and P_v are the respective coordinates in enrollment and verification images of the keypoints. The quality of the homography model is then evaluated according to how many of the 50 keypoint pairs fit the model. Each pair that fits the homography model is considered an inlier. The process iterates to calculate and evaluate homographies from different sample points, and the homography with the highest number of inliers is returned as the best fit for the data.

$$P_v = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \times P_e \quad (5.1)$$

5.3.3.2 Projection and Scoring

Using the enrollment and verification keypoints, and the homography between them, we compute a score that indicates how many of the enrolled keypoints have good matches in the set of verification keypoints. An enrolled keypoint is considered to have a good match if there exists a verification keypoint that satisfies two conditions: (1) it is highly similar to the enrolled keypoint, and (2) it is at the position where the enrolled keypoint should be found in the verification image. The first condition is formalized as a requirement of being the nearest neighbor in feature space to the enrolled keypoint. The second condition is formalized as a requirement of being within 2 pixels of the location where the homography predicts the enrolled keypoint to be in

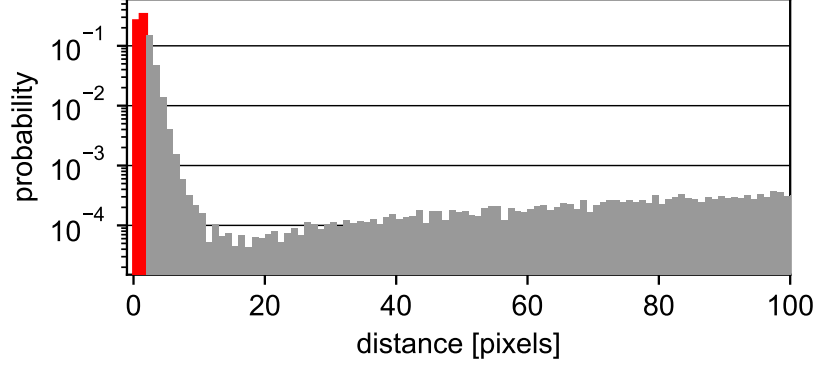


Figure 5.5: Pixel distance between the expected location of a keypoint (according to homography) and the location of its nearest neighbor in feature space. The spike at left shows points for which the nearest neighbor is found in the expected location. The points that are sufficiently close to be counted as inliers are the ones colored red.

the verification image. This ensures that matched features are not only similar, but also geometrically consistent with relative positions of the enrolled keypoints. Fig. 5.5 shows the pixel distance between the homography projection of an enrolled keypoint and the location of the verification keypoint that is its nearest neighbor in feature space. The data is collected from 100 different verification trials. The peak at left indicates that the nearest neighbor is often found within two pixels of the location predicted by the homography. These points are the inliers.

Fig. 5.6 shows examples of keypoint matching from verification. The matching succeeds even when the verification image is rotated and at a different scale from the orientation of the same chip at enrollment. Each line on the figure shows the correspondence between an enrolled keypoint and a matching keypoint found on the package during verification.

5.4 Evaluation

We evaluate the COUNTERFOIL system using experiments on populations of two plastic dual in-line package (PDIP) chips. The first is an Alliance Memory AS6C6264-55PCN [7], which is a 64kb SRAM in a 28-pin PDIP (surface size 35.6mm \times 15.2mm)

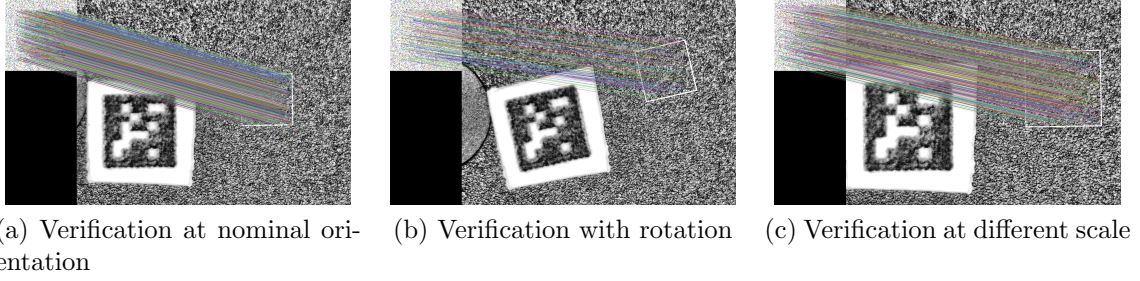


Figure 5.6: Three examples of matching between enrollment keypoints (square in upper left) and verification image of the same chip package instance, where the verification image differs in zoom and orientation. White square on chip package is the identified region of interest for verification. Each line corresponds to a keypoint match from enrollment to verification (Sec. 5.3.3.2).

that is rated for 0°C to 70°C temperature range. The second is a Microchip Technology 23LC1024 [80], which is a 1Mb SRAM in an 8-pin PDIP ($9.2\text{mm} \times 6.4\text{mm}$) that is rated for -40°C to 85°C . Images are collected using two instances of two different camera models. The two ViTiny UM12 cameras [113] cost \$390 each, have 5MP sensors, and computer-controlled focus through software. The two MustCam UM012C cameras cost \$40 each, have 5MP sensors, and manual focus by turning a dial. Our collection of chips and cameras are shown in Fig 5.7.

In our evaluation we use 52 instances of chip model AS6C6264 and 40 instances of chip model 23LC1024. Chips packaged in the same mold are identified by the mold marking on the package. Our dataset has several chips packaged from the same mold: 5 pairs, 9 multiples in chip model AS6C6264 and 14 pairs in chip model 23LC1024. Each chip instance is enrolled to the database using one camera, and then verified using the other camera of the same model. Enrollment and verification is repeated 3 times for each chip, comprising a total of 528 images taken with ViTiny and MustCam.

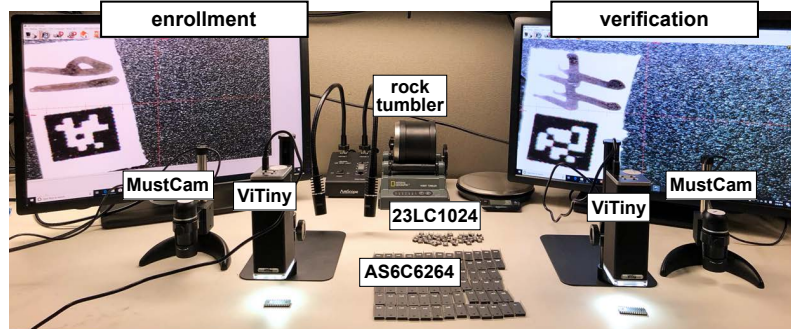


Figure 5.7: Experimental setup. Left side of workbench used for enrollment, right side used for verification. Separate camera are used for enrollment and verification. Middle of image shows the population of chips with labels affixed.

5.4.1 Package Authentication

Package authentication is performed by matching verification image features with enrolled ones as described in Sec. 5.3. Fig. 5.8 shows in green the cumulative distribution function (CDF) of the number of inliers (matched keypoints) from the dataset of enrolled and verification chip images using our system. Fig. 5.8 also shows in red the CDF of inliers for mislabeled packages. In these cases, the program is modified to ignore the identity encoded on the label, and to fetch from the database the enrolled keypoints of another, randomly selected chip instance of the same model. 5,000 such comparisons are performed. This CDF represents what a counterfeiter might achieve by randomly swapping labels. We also consider the strongest counterfeiter that has an exact duplicate of the mold used to produce the enrolled chip, and he copies the label for the legitimate enrolled chip onto his counterfeits created from the same mold. The lines in blue show the number of inliers that the counterfeit would be able to achieve in this permissive setting. Even if the attacker has the same mold used to produce an enrolled chip, the counterfeits that can be created with the mold typically still have significantly fewer inliers than the enrolled chip.

The verifier’s decision to accept or reject a package is made according to whether the number of matched enrollment keypoints exceeds a threshold. A higher thresh-

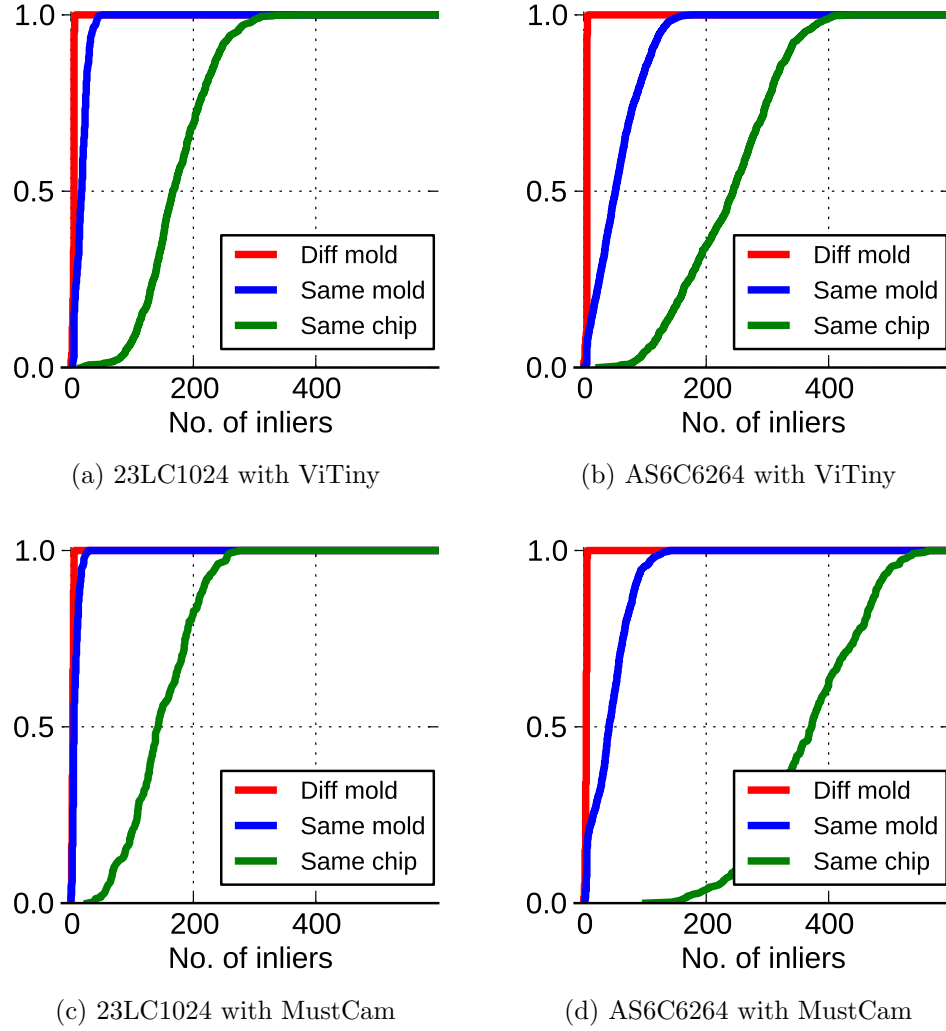


Figure 5.8: CDF of number of inliers using each model of camera.

old is a more selective determination of authenticity. Higher thresholds reduce false positives (counterfeit chips being accepted as authentic), but can also reduce true positives (legitimate chips being accepted as authentic). Receiver operating characteristic (ROC) curves are plots that show the achievable tradeoffs between excluding false positives and keeping true positives. An ROC curve is created by sweeping the acceptance threshold, and at each threshold value evaluating the number of true and false positives. A true positive always refers to a case where the enrolled and verified chip are the same instance with the same label, but we use two different notions of a

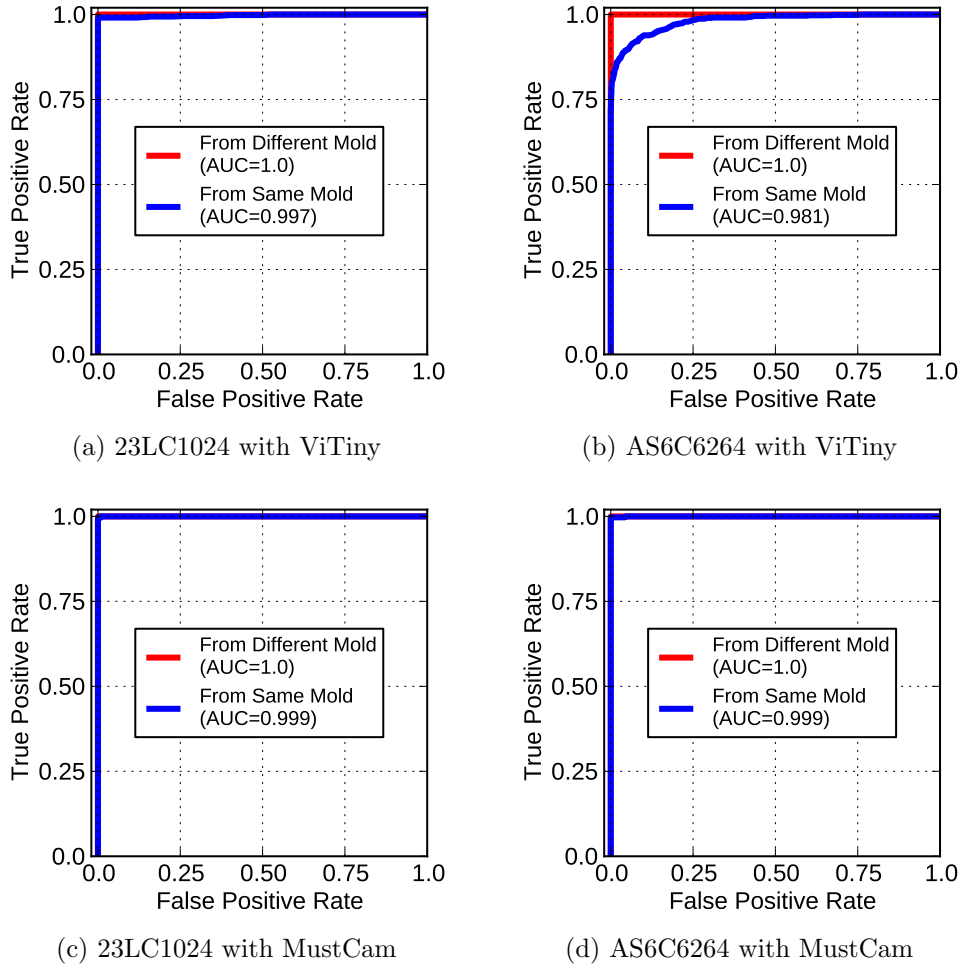


Figure 5.9: Receiver Operating Characteristic curves show ability to distinguish enrolled chips from other chips created from a different mold than the enrolled chip, or from the same mold that produced the enrolled chip.

false positive. The first case is when the chip being verified is a counterfeit that differs from its labeled identity, and the identity on the label is not enrolled to a chip from the same mold as the counterfeit. The second case is when the counterfeit chip has a label used to enroll another chip from the same mold as itself. The ROC curves are shown in Fig. 5.9. For both models of chip and both models of camera, we are able to distinguish perfectly (100% true positives at 0% false positives) between a legitimate chip being verified and a counterfeit chip that is created from a different mold. Even in the extreme case where the counterfeiter has the same mold (from the packaging

house) used to create the enrolled chip, it is possible to detect the counterfeits while still keeping a high rate of true positives. The worst case is AS6C6264 with ViTiny camera (Fig. 5.9b), where it is still possible to accept 90% of legitimate chips while rejecting 90% of counterfeits that are created from the exact same mold. We will show later in the paper that this performance can be further improved by higher quality images.

5.4.2 Runtime

Verifying provenance of packages should not slow manufacturing (for enroller) or integration (for verifier). The verification process is more computationally intensive than enrollment, and certain target applications for verification may impose stringent latency requirements. For example, we envision that one application is integration with a pick-and-place machine, which removes chips from feeder reels and places them appropriately onto printed circuit board pads for reflow soldering. Single head pick-and-place machines from a leading manufacturer place between 1,800 and 5,000 parts per hour [76], which corresponds to handling each part for 720ms to 2s. Fig. 5.10 shows that package verification can be performed at production speed, as our system is able to authenticate each instance within 150 ms on an Intel Xeon CPU E5-2690. The runtime can be further reduced to meet even tighter latency requirements by enrolling a smaller number keypoints for each chip. Fig. 5.10 shows how runtime scales with the size of ROI at a constant keypoint density, and shows the breakdown of runtime according to image processing function.

5.4.3 Practicality and Costs

The COUNTERFOIL methodology is compatible as an add-on to existing supply chains, and the cost at scale should be significantly less than one cent per chip. Chip verifiers can use the inexpensive camera models from our experiments, and perform processing on dedicated or shared computers. Given that verification would

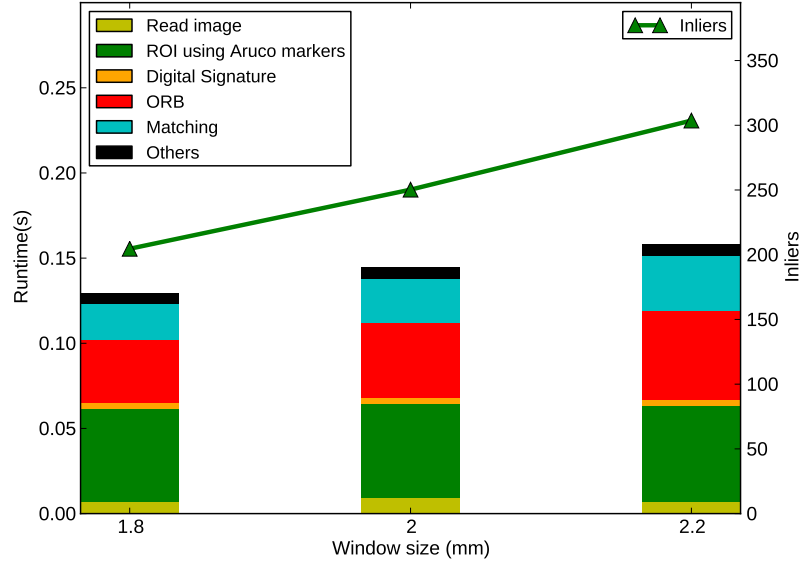
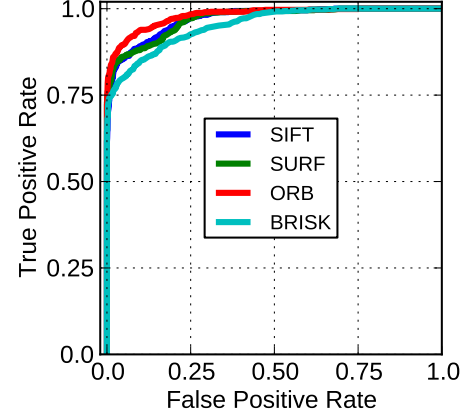


Figure 5.10: Runtime of verification procedure, broken down by processing task, for different sizes of ROI. Keypoint density is held constant at $1,000/mm^2$. The increase in keypoints for the larger ROI results in a higher runtime, but also increases the number of matching points that are found. Runtime can be traded against accuracy by adjusting the ROI size.

likely be performed at PCB assembly houses, the small cost of the camera would be insignificant, especially when amortized over a large number of boards being produced. The labels affixed to the chips cost \$0.30 per sheet, and we print 1024 markers per sheet, for a per-unit cost of \$0.0003 per label. The enrolled data for each chip is 1 MB, which at current hard-drive prices of \$0.03 per GB corresponds to a per-unit cost of \$0.00003 for storing the data. Affixing markers to each chip is currently a manual and time-consuming process. At scale we imagine that per-chip labels could be replaced by labels on part reels, or other ways of communicating a purported identity for the parts that would be used to access the signed enrollment records. In that case, the ROI would be identified based on image recognition of package surface instead of the markers. The low barriers to adoption of COUNTERFOIL are simply convincing a packaging house to deploy this technology, and establishing keys for signing and verifying chips. Even if only a small fraction of purchasers would

Table 5.1: Quantitative comparison of different feature-detecting methods. Plot at right shows the ROC plot from which the area-under-curve is computed. All four algorithms are configured to use 1,000 keypoints per mm^2 for this comparison.

Algorithm	Avg. Inliers		Area Under Curve	Run Time [s]
	Same Chip	Same Mold		
SIFT	570	178	0.971	0.215
SURF	470	100	0.970	0.211
ORB	236	56	0.980	0.064
BRISK	215	53	0.953	0.432



verify their chips using the available information, this should increase the risk of detection for distributors that traffic in possible counterfeits. The more significant barrier to adoption is perhaps the possibility that superficial cosmetic damage to parts could cause them to become untrusted, representing a monetary loss and a harm to branding.

5.4.4 Algorithm Difference

Tab. 5.1 compares the runtime and authentication performance of four popular algorithms for feature extraction and matching. While all of the algorithms are suitable, we find ORB to perform best, and have thus chosen it for our work. In particular, the speedup of ORB comes largely from its compatibility of using locality-based hashing to identify near neighbors, without using the k-nearest neighbor search which is the most time consuming operation in the other algorithms.

5.4.5 Camera Differences

Because enrollment and verification are performed using different camera instances, ability to match features may be impacted by differences in the lens, lighting,

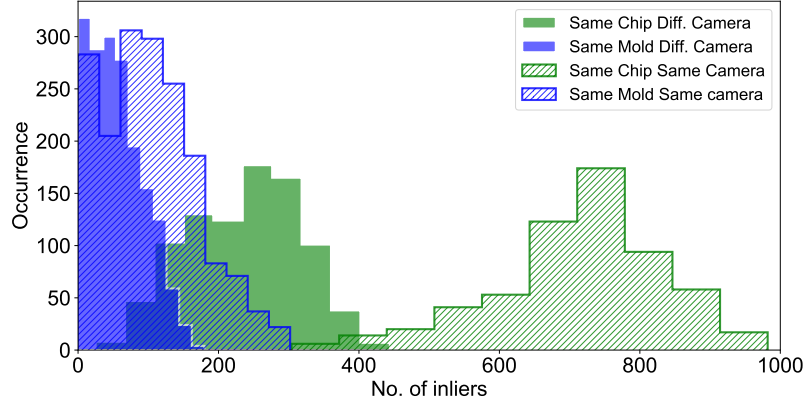


Figure 5.11: Histograms showing increase in number of inliers in AS6C6264 SRAM when same ViTiny cameras are used for both enrollment and verification.

or the sensor array [73] of the cameras. To explore this further, we now evaluate how the matching performance changes in the unrealistic scenario of using the same ViTiny camera instance for both enrollment and verification of AS6C6264 chips, which was the most challenging authentication case in the prior experiments (see Fig 5.9b). Fig. 5.11 shows that using a consistent camera causes the number of inliers to increase, both in the case of same-chip comparisons and same-mold comparisons. The same-chip comparisons have a larger increase, and the overlap between the two distributions is reduced, implying capability for better authentication performance. This result reveals the presence of some detrimental camera variations that are being overcome in our realistic authentications that use different camera instances for verification and enrollment.

5.4.6 Varying Magnification and Lighting

Fig. 5.12 shows results under different magnification and lighting conditions using the ViTiny camera with the AS6C6264 chips using a smaller dataset with 10 chip instances. The approach is largely unaffected by lighting changes, but changing the magnification from enrollment to verification has some impact on the number of inliers.

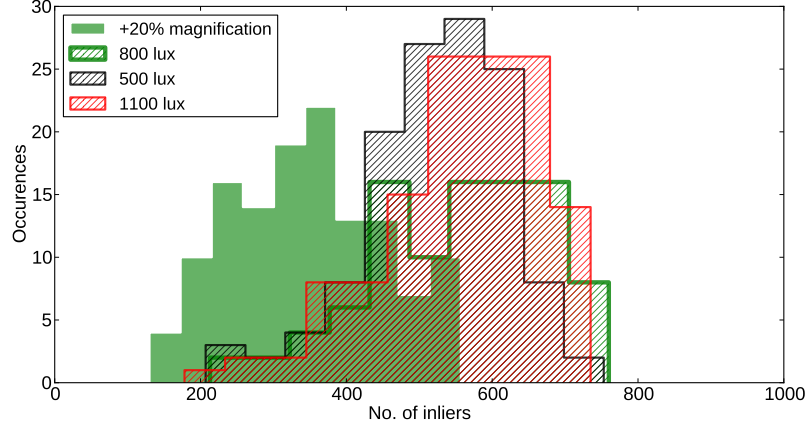


Figure 5.12: Histogram of inliers in AS6C6264 SRAM under two alternative lighting intensities (nominal is 800 lux) and one alternative zoom.

5.5 Further Investigation of Fingerprints

In this section we deviate from our standard system to investigate package fingerprint properties that cannot easily be evaluated within the overall system. In particular, for different reasons, experiments in this section define the ROI in a way that doesn't rely on affixed labels. Instead of defining the center of the ROI as being at position $\langle r, \theta \rangle$ relative to the marker (see Fig. 5.4), the center of the ROI is here defined as a pixel in the center of the image. To ensure that the same area of the chip is always imaged, the chip is aligned carefully to the camera. Aside from lacking markers, the image processing performed is as described in Sec. 5.3.

5.5.1 Testing Resilience of Fingerprints

The fingerprints should be robust enough to withstand wear that occurs when IC packages are jostled and handled during distribution. We use various time durations in a hobbyist rock tumbler to impart controllable amounts of wear on chips. After enrollment, chips are placed alone in the rock tumbler with 45mL of water and 5g of 60-grit silicon carbide, which is the coarsest grit used in rock tumbling. The tumbler barrel is washed out between experiments, and each trial uses new grit and clean water. After tumbling, the chip is removed, rinsed under a faucet, dried and imaged

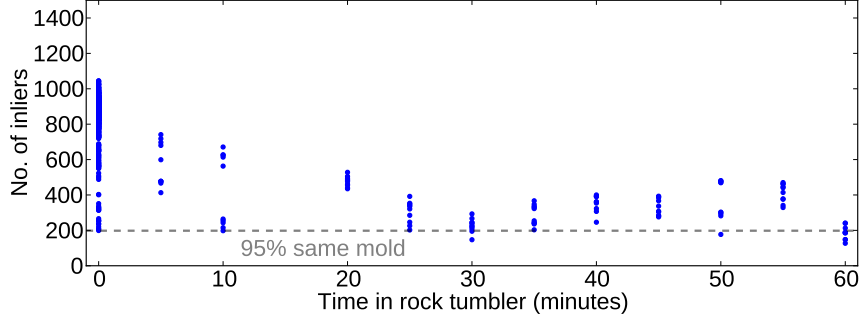
for verification. The prototype adhesive labels do not survive the rock tumbler, so the ROI in the images is instead found by careful alignment of the chip under the camera.

Fig. 5.13 shows the degradation in number of inliers for chips after different amounts of time in the tumbler. The plot shows a slow decrease in the number of inliers after tumbling with a few hundred inliers left after an hour in the tumbler. The dashed line on the plot shows the acceptance threshold that has a 95 percent probability of rejecting a different chip from the same mold. In other words, an attacker that has obtained the same mold and produced new chips from it will have only a 5% of exceeding this threshold and thereby succeeding in forgery. Even after significant wear, most authentication trials from the legitimate chip are able to exceed this value.

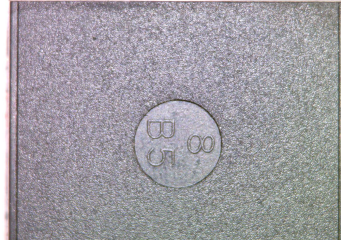
Figs. 5.13b and 5.13c show package surfaces before and after 1 hour in the tumbler. Note that these images are illustrative; they use a different magnification from the results in Fig. 5.13a and include the corners of the chip where the wear is most noticeable, instead of showing only the ROI where the wear is less apparent. We also tested the effect of temperature by heating the chips to 170°C for an hour in a thermal chamber, but saw no change in the number of inliers.

5.5.2 Testing Fingerprint Uniqueness

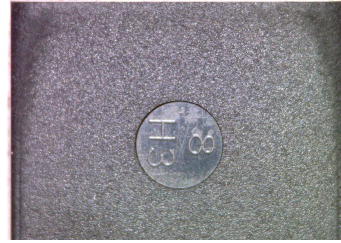
Any complex physical object has some combination of minute features that are unlike all other instances of the same object. Given that molded integrated circuit packages are heterogeneous mixtures of particles, they are certain to be unique in this trivial, physical, sense. However, for authentication the relevant question is whether there is a uniqueness that is observable and stable at the scale of our imaging. In studying uniqueness, we pay special attention to chips that are produced from the same mold. Fortunately, each chip bears a mold mark that is imprinted in a circle on



(a) Reduction in inliers after wear in rock tumbler



(b) Chip before tumbler



(c) Chip after 1 hour

Figure 5.13: Reduction in inliers for chip AS6C6264 after spending time in rock tumbler. Images of chip are included to give a sense of the amount of wear caused.

the underside of the chip. The mold mark, as is visible in Fig. 5.13b, gives a code of one letter and two numbers. The marks are used for traceability within the packaging facility, so that problematic molds can be identified. Our experiments confirm that chips with the same mark are from the same mold, as they show a distinct similarity according to our analysis, and in fact a similar texture can be observed at high magnification.

5.5.2.1 Scoring under Controlled Alignment

Experiments that use imprecisely placed labels to define the ROI of each chip cannot definitively show whether package fingerprints are unique. Two packages that are identical would appear unique if their labels are placed in such a way that their ROIs are disjoint regions of the package surface. We again avoid relying on markers and perform experiments in which ROI is based on chip alignment underneath the camera. Fig. 5.14 shows the result. Different chip instances from the same mold do

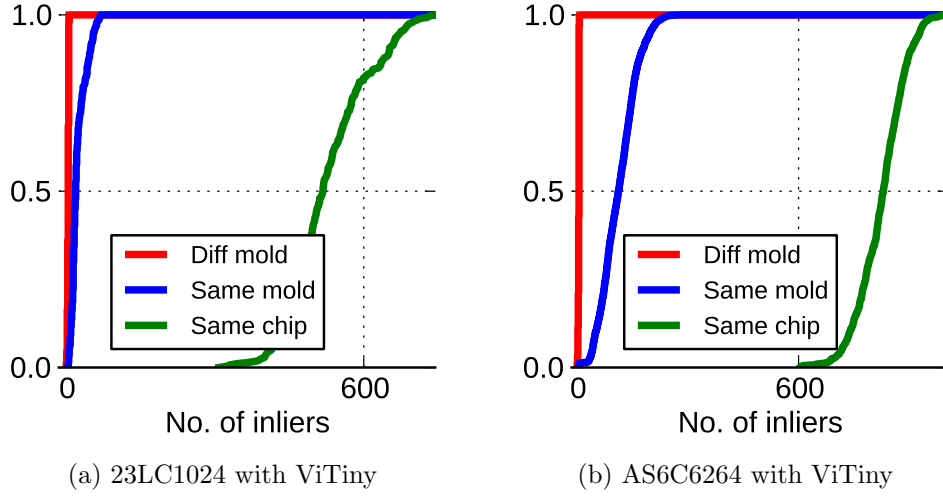
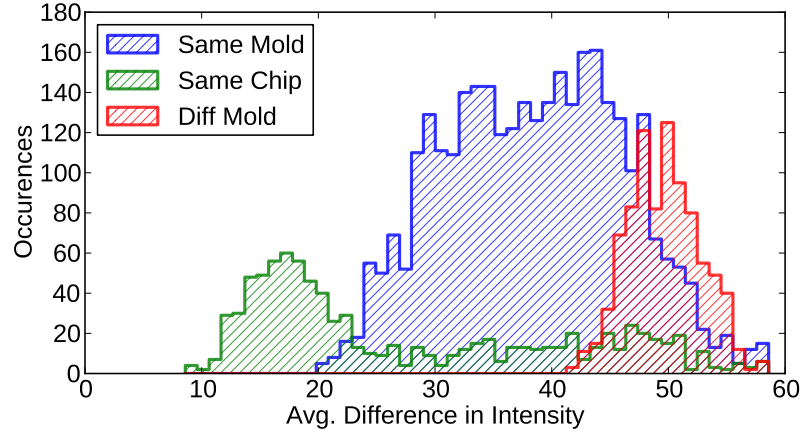


Figure 5.14: Inlier CDFs for SRAMs under controlled alignment.

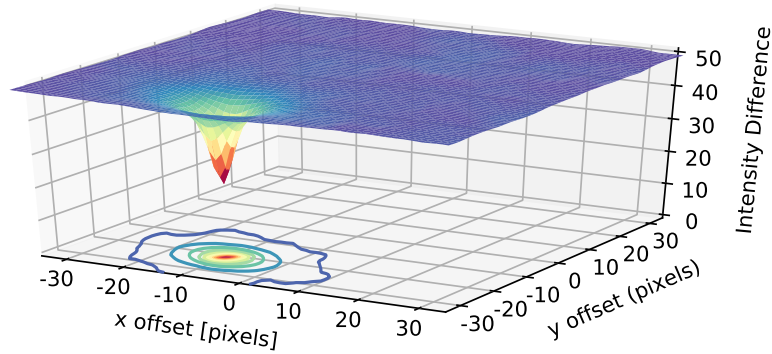
show similarity, but it is smaller than the similarity between two images of the same chip. In chip type AS6C6264, the highest score between any two images of different chips from the same mold is 277 inliers, whereas the lowest score between any two images of the same chip is 603 inliers; the means are 113 and 825 respectively. The clear difference in scores for same-mold and same-chip comparisons is significant, as it shows that the mold surface texture is not entirely responsible for the fingerprints. Even if an adversary were able to perfectly reproduce (or steal) the mold, they will be unable to create high quality forged packages with it.

5.5.2.2 PUF-like evaluation using Pixel Intensity

We also consider evaluating similarity of package fingerprints using a standard Physically Unclonable Function(PUF)-like scheme rather than the computer vision based techniques used in COUNTERFOIL. As standard PUF metrics [75, 74] based on Hamming distance are not directly applicable in this setting, distance comparisons between enrollment and verification images are made by comparing the 8-bit pixel intensities of the two ROIs on a pixel-by-pixel basis, which is analogous to comparing responses from weak PUFs on a bit-by-bit basis.



(a) Difference in Pixel Intensity



(b) Alignment

Figure 5.15: PUF-like evaluation on raw pixel intensity data.

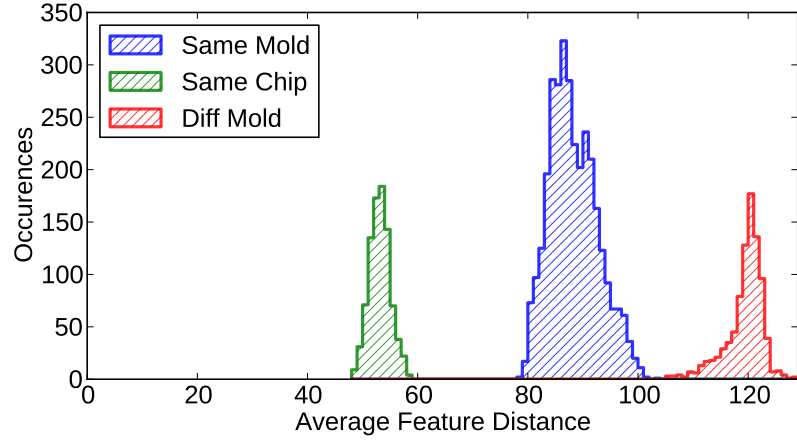
The major challenge in making this comparison is that, unlike in digital PUFs, when comparing images there is no ground truth about which pixel in the verification image should be compared against which pixel in the enrollment image. Even if the package appears identical in the two images, the pixel-by-pixel comparison will only show the similarity if the two images have pixel-accurate alignment. Aside from requiring pixel-accurate alignment in the X and Y directions, rotation and scale variance additionally cannot be tolerated. Still, with some difficulty, we can partially overcome these challenges to make a pixel-by-pixel comparison. To make the comparison, we start from images taken using controlled alignment. A brute-force search is then performed to find the X and Y offset that best aligns the images, as seen

in Fig. 5.15b. Only when the alignment is correct to within a few pixels does the similarity between the images become apparent. The need to perform brute force search for alignment increases runtime to 10s per comparison, which is hundreds of times slower than COUNTERFOIL, and still unable to handle any change to rotation or scale. The results from making hundreds of comparisons in this manner are shown in Fig. 5.15a. In some cases, presumably due to rotation or scale, the similarity between the same-chip images cannot be found using pixel-by-pixel comparisons. This result confirms that the package features can with some difficulty be observed in a PUF like way, but also shows that pixel-by-pixel comparisons are not well-suited to this task relative to the computer vision approach.

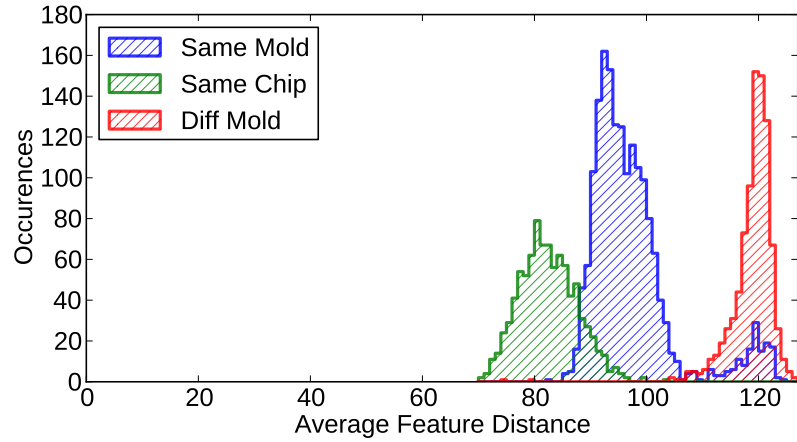
5.5.2.3 PUF-like evaluation using Feature Distance

In COUNTERFOIL, the number of matches that we compute as inliers is based on both feature similarity, and the geometric relationship of the features on the package surface, as matched keypoints from enrollment and verification must be related by a homography. One might also consider evaluating similarity of the features in corresponding positions of two chip packages, similar to Hamming Distance between corresponding bits in a PUF circuit. In this case, the computer vision approach is being used to align the enrollment and verification keypoints, but after alignment is decided the corresponding features are scored according to their similarity in feature space instead of their pixel intensity.

Fig. 5.16 shows the average distance, in feature space, between features having positional correspondence defined by computed homography. In a highly controlled setting of careful alignment, lighting and single camera, the same package can be distinguished from packages created from the same mold, as shown by the separation between the feature distances in Fig. 5.16a. However, in the general setting which contains typical image quality variations, the same chip distribution is shifted to



(a) Controlled setting



(b) Uncontrolled setting

Figure 5.16: Average distance in feature space for same-position keypoint pairs.

the right leading to a slight overlap with the same mold distribution as shown in Fig. 5.16a. An absolute feature distance threshold to distinguish between chips from same mold is therefore not robust to image quality variations. COUNTERFOIL aims to avoid this limitation by using feature similarity ranking (nearest neighbors) instead of an absolute distance threshold.

5.5.3 Additional Package Types

To further validate package surface fingerprints, we conduct experiments with 10 additional types of circuit packages. As before, one ViTiny camera is used for

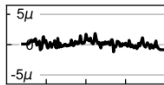


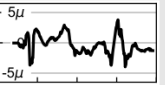
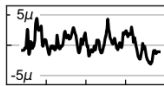


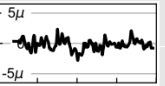
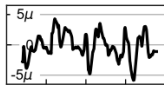


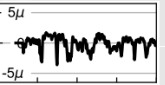
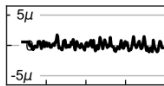


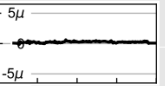
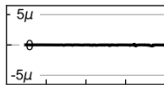


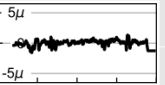
Surface Map	Example Image	Chip Name	Package	Same Chip	Different Chip	Area (mm^2)	Example Image	Surface Map
		W25Q80EWUXIETR	23-SOT	38.60	3.02	0.454		
		TSV524IQ4T	16-QFN	42.90	4.10	0.315		
		MX25V4006EM11-13G	8-SOIC	58.83	3.77	0.454		
		24LC32A-I/MS	8-MSOP	344.35	3.98	2		
		CY7C1353G-100AXC	100-TQFP	280.77	4.66	2		
		ADG419TQ	14-CDIP	358.39	3.89	2		
		ADP125ACPZ-R7DKR-ND	8-LFCSP	18.28	3.22	0.315		
		W25Q80EWUXIE TR	8-USON	12.27	2.08	0.201		
		FAN53540UCX	20-WLCSP	3.93	1.76	0.315		
		2N3440	TO-39	0	0	2		

Figure 5.17: Evaluation of package surface fingerprints across a range of package types.

enrollment, and a second for verification. We use 5 instances of each chip, and from each instance collect 5 enrollment and 5 verification images. Note that, among the molded packages in this secondary population, we don't have any chips that appear to be from the same mold.

Fig. 5.17 summarizes the results of the experiment. Because many of the packages are quite small, and we want to use an unmarked area of the package surface as the fingerprint, in some cases the enrolled area of the surface is smaller than $2mm^2$. ROI is identified by manual chip alignment under the camera, as many of the packages are impractically small for the crude adhesive markers used in our prototype demonstration. The table gives for each chip an example image with the ROI marked by a square. To give a sense of the surface structure of each package model, we plot within the table the deviation from nominal surface height along an arbitrary 0.9mm trace of the surface; this data is collected with the same Zygo Nexview 3D optical surface profiler used to generate Fig. 5.3a.

The significant distance between the average number of inliers for same chip and different chip comparisons implies that it may be possible to authenticate most of the plastic packages by their fingerprints, although further experiments would be needed to give confidence. Interestingly, based on this preliminary data, the ceramic package (14-CDIP) also appears to be highly identifiable. Two packages that are notably unsuitable for the style of package fingerprinting used in this paper are the final two entries in the table – the TO-39 metal can package and 20-WLCSP wafer-level package. In these two cases, the reflective surfaces cause very few keypoints to be extracted from the image, and the extracted keypoints do not match well between enrollment and verification.

5.6 Summary

In this chapter we have presented COUNTERFOIL, a system that verifies provenance by extracting unique fingerprints from surface features of integrated circuit packages imaged using inexpensive cameras. The work is a low-cost strategy that can help to address the significant problem of counterfeit integrated circuits which results in billions of dollars of losses each year. Our approach enrolls unique features of each chip during packaging, and requires no chain-of-custody. During verification features are matched against cryptographically signed enrollment records. We’ve demonstrated the approach to work on a large population of two different chips, have used different models of low-cost microscope cameras, and have evaluated resiliency of fingerprints. Crucially, we’ve shown that even an adversary possessing an exact duplicate of the mold used to produce a chip’s package will not be able to create a high-quality counterfeit of the chip.

CHAPTER 6

CONCLUSION

In this thesis we have addressed the important problem of security of ICs in the IoT application space, with a specific focus on block ciphers. When implementing block ciphers there is a choice of implementation style depending on design and resource constraints. For low latency requirements, our Combinational Checkpointing scheme (Ch. 2) is shown to be the most efficient way to implement unrolled implementations of block ciphers but the 2-10x area costs of unrolling are unappealing. In low area implementations, our novel lightweight AES architecture (Ch. 3) based on register renaming greatly reduces inefficiencies in data movement and clocking, making it twice as efficient as state-of-the-art. Side channel resilience is also improved by adding shuffling capability to randomize sub-round operations of the AES algorithm. We successfully taped out our designs in a commercial 16nm FinFET technology (Ch. 4) and at 0.55 pJ/bit our renaming architecture is to the best of our knowledge the most efficient one to date, making it an attractive candidate for low power applications.

We also present a novel cost-effective methodology, denoted as COUNTERFOIL, to tackle the serious problem of IC counterfeiting (Ch. 5). COUNTERFOIL leverages the variability in IC packaging to extract unique fingerprints from package surface texture and uses them to verify chip provenance. Our technique is shown to work with different types of plastic packages and is resilient to imaging conditions and, to some degree, wear-and-tear. The low cost and ease of integration make COUNTERFOIL a compelling solution to ensure supply chain security.

LIST OF PUBLICATIONS

- **SN Dhanuskodi**, D Holcomb, *Enabling Microarchitectural Randomization in Serialized AES Implementations to Mitigate Side Channel Susceptibility*, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2019.
- C Ramesh, S Patil, **SN Dhanuskodi**, G Provelengios, S Pillement, D Holcomb, R Tessier, *FPGA Side Channel Attacks without Physical Access*, IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018.
- **SN Dhanuskodi**, D Holcomb, *An improved clocking methodology for energy efficient low area AES architectures using register renaming*, IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp 1-6, 2017.
- **SN Dhanuskodi**, D Holcomb, *Techniques to Reduce Switching and Leakage Energy in Unrolled Block Ciphers*, IEEE Transactions on Computers, Aug. 2017.
- **SN Dhanuskodi**, D. Holcomb, *Energy Optimization of Unrolled Block Ciphers using Combinational Checkpointing*, RFIDSec: 12th Workshop on RFID and IoT Security, published as post-proceedings in Springer Lecture Notes in Computer Science (LNCS), pp 47-61, 2016.
- **SN Dhanuskodi**, S. Keshavarz, D. Holcomb, *LLPA: Logic State Based Leakage Power Analysis*, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp 218-223, 2016.

BIBLIOGRAPHY

- [1] Amazon F1 web site. <https://aws.amazon.com/ec2/instance-types/f1/>.
- [2] Ncsu free pdk 45. <http://www.eda.ncsu.edu/wiki/FreePDK45:Contents>.
- [3] Crypto++ Library 8.1.0, Feb 2019. <https://www.cryptopp.com/>.
- [4] A. Putnam et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *ISCA* (June 2014), pp. 13–24.
- [5] Akkaya, N. E. C., Erbagci, B., and Mai, K. Secure chip odometers using intentional controlled aging. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (April 2018), pp. 111–117.
- [6] Alioto, M., Bongiovanni, S., Djukanovic, M., Scotti, G., and Trifiletti, A. Effectiveness of leakage power analysis attacks on DPA-resistant logic styles under process variations. *IEEE Transactions on Circuits and Systems I: Regular Papers* 61, 2 (Feb 2014), 429–442.
- [7] Alliance Memory Inc. AS6C6264: 8k x 8bit Low Power CMOS SRAM, 2017. https://www.alliancememory.com/wp-content/uploads/pdf/Alliance%20Memory_64K_AS6C6264v2.0July2017.pdf.
- [8] Asadizanjani, Navid, Dunn, Nathan, Gattigowda, Sachin, Tehranipoor, Mark, and Forte, Domenic. A database for counterfeit electronics and automatic defect detection based on image processing and machine learning. *ISTFA*, Nov (2016).
- [9] Banik, S., Bogdanov, A., Regazzoni, F., Isobe, T., Hiwatari, H., and Akishita, T. Round gating for low energy block ciphers. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (May 2016), pp. 55–60.
- [10] Banik, Subhadeep, Bogdanov, Andrey, and Regazzoni, Francesco. Exploring energy efficiency of lightweight block ciphers. In *International Conference on Selected Areas in Cryptography* (2015), Springer, pp. 178–194.
- [11] Batina, Lejla, Das, Amitabh, Ege, Barış, Kavun, Elif Bilge, Mentens, Nele, Paar, Christof, Verbauwhede, Ingrid, and Yalçın, Tolga. Dietary recommendations for lightweight block ciphers: Power, energy and area analysis of recently developed architectures. In *Radio Frequency Identification*. Springer, 2013, pp. 103–112.

- [12] Bay, Herbert, Tuytelaars, Tinne, and Van Gool, Luc. Surf: Speeded up robust features. In *Computer Vision – ECCV 2006* (Berlin, Heidelberg, 2006), Aleš Leonardis, Horst Bischof, and Axel Pinz, Eds., Springer Berlin Heidelberg, pp. 404–417.
- [13] Beaulieu, Ray, Shors, Douglas, Smith, Jason, Treatman-Clark, Stefan, Weeks, Bryan, and Wingers, Louis. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. <http://eprint.iacr.org/2013/404>.
- [14] Benini, L., Micheli, G. De, Macii, A., Macii, E., Poncino, M., and Scarsi, R. Glitch power minimization by selective gate freezing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8, 3 (June 2000), 287–298.
- [15] Benson, Richard C, Farrar, Dawnielle, and Miragliotta, Joseph A. Polymer adhesives and encapsulants for microelectronics applications. *Johns Hopkins APL Technical Digest* 28, 1 (2008), 58.
- [16] Bergman, T. D., Manager, C. P., and Liszewski, K. T. Battelle barricade: A nondestructive electronic component authentication and counterfeit detection technology. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)* (May 2016), pp. 1–6.
- [17] Bertoni, Guido, Macchetti, Marco, Negri, Luca, and Fragneto, Pasqualina. Power-efficient ASIC synthesis of cryptographic sboxes. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI* (New York, NY, USA, 2004), GLSVLSI '04, ACM, pp. 277–281.
- [18] Biryukov, Alex, and Wagner, David. Slide attacks. In *Proceedings of the 6th International Workshop on Fast Software Encryption* (London, UK, UK, 1999), FSE '99, Springer-Verlag, pp. 245–259.
- [19] Boemo, Eduardo, Oliver, Juan P., and Caffarena, Gabriel. Tracking the pipelining-power rule along the FPGA technical literature. In *Proceedings of the 10th FPGAworld Conference* (New York, NY, USA, 2013), FPGAworld '13, ACM, pp. 9:1–9:5.
- [20] Bouesse, G. F., Renaudin, M., Witon, A., and Germain, F. A clock-less low-voltage AES crypto-processor. In *Proceedings of the 31st European Solid-State Circuits Conference, 2005. ESSCIRC 2005.* (Sept 2005), pp. 403–406.
- [21] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
- [22] Brier, Eric, Clavier, Christophe, and Olivier, Francis. *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*. 2004, ch. Correlation Power Analysis with a Leakage Model, pp. 16–29.

- [23] Calhoun, B. H., Wang, A., and Chandrakasan, A. Modeling and sizing for minimum energy operation in subthreshold circuits. *IEEE Journal of Solid-State Circuits* 40, 9 (Sept 2005), 1778–1786.
- [24] Canright, David. A very compact S-box for AES. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2005), Springer, pp. 441–455.
- [25] Christopher Henderson. Transfer Molding, 9 2012. In InfoTracks Semitracks Monthly Newsletter; Available: <http://www.semitracks.com/newsletters/september/2012-september-newsletter.pdf>.
- [26] Cobb, W. E., Laspe, E. D., Baldwin, R. O., Temple, M. A., and Kim, Y. C. Intrinsic physical-layer authentication of integrated circuits. *IEEE Transactions on Information Forensics and Security* 7, 1 (Feb 2012), 14–24.
- [27] Czajkowski, T. S., and Brown, S. D. Using negative edge triggered FFs to reduce glitching power in FPGA circuits. In *2007 44th ACM/IEEE Design Automation Conference* (June 2007), pp. 324–329.
- [28] DeJean, Gerald, and Kirovski, Darko. *RF-DNA: Radio-Frequency Certificates of Authenticity*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 346–363.
- [29] Dhanuskodi, Siva Nishok, and Holcomb, Daniel. Energy optimization of unrolled block ciphers using combinational checkpointing. In *RFIDSec 2016: 12th Workshop on RFID and IoT Security, 2016* (Dec 2016).
- [30] Dhanuskodi, Siva Nishok, and Holcomb, Daniel. Techniques to reduce switching and leakage energy in unrolled block ciphers. *IEEE Transactions on Computers* (2017), 1–1.
- [31] Dhanuskodi, Siva Nishok, and Holcomb, Daniel. Enabling microarchitectural randomization in serialized AES implementations to mitigate side channel susceptibility. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2019).
- [32] Dhanuskodi, Siva Nishok, and Holcomb, Daniel E. An improved clocking methodology for energy efficient low area AES architectures using register re-naming. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)* (July 2017).
- [33] Dhanuskodi, Siva Nishok, Keshavarz, Shahrzad, and Holcomb, Daniel. LLPA: logic state based leakage power analysis. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2016).
- [34] Digilent. Cmod A7: Breadboardable Artix-7 FPGA Module. <https://store.digilentinc.com/cmod-a7-breadboardable-artix-7-fpga-module/>.

- [35] Dumpala, Naveen Kumar, Patil, Shivukumar B, Holcomb, Daniel, and Tessier, Russell. Energy efficient loop unrolling for low-cost FPGAs. In *Field-Programmable Custom Computing Machines (FCCM), 2017 IEEE 25th Annual International Symposium on* (2017), IEEE, pp. 117–120.
- [36] Dworkin, Morris J. SHA-3 standard: Permutation-based hash and extendable-output functions. Tech. rep., 2015.
- [37] Feldhofer, Martin, Dominikus, Sandra, and Wolkerstorfer, Johannes. Strong authentication for RFID systems using the AES algorithm. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2004), Springer, pp. 357–370.
- [38] Fischler, Martin A., and Bolles, Robert C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (June 1981), 381–395.
- [39] Galton, Francis. *Fingerprint directories*. Macmillan and Company, 1895.
- [40] Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F.J., and Marín-Jiménez, M.J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogn.* 47, 6 (June 2014), 2280–2292.
- [41] Gassend, B, Clarke, D, and Van Dijk, M. Silicon physical random functions. In *Proceedings of the IEEE Computer and Communications Society* (2002).
- [42] Giechaskiel, Ilias, Rassmussen, Kasper B., and Eguro, Ken. A robust covert channel on FPGAs based on long wire delays. *CoRR abs/1611.08882v2* (2017).
- [43] Giorgetti, Jacopo, Scotti, Giuseppe, Simonetti, Andrea, and Trifiletti, Alessandro. Analysis of data dependence of leakage current in CMOS cryptographic hardware. In *Proceedings of the 17th ACM Great Lakes Symposium on VLSI* (2007), GLSVLSI '07, pp. 78–83.
- [44] Guajardo, J, Kumar, S, Schrijen, GJ, and Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems* (2007).
- [45] Guin, Ujjwal, Huang, Ke, DiMase, Daniel, Carulli, John M, Tehranipoor, Mohammad, and Makris, Yiorgos. Counterfeit integrated circuits: a rising threat in the global semiconductor supply chain. *Proceedings of the IEEE* 102, 8 (2014), 1207–1228.
- [46] Guin, Ujjwal, Zhang, Xuehui, Forte, Domenic, and Tehranipoor, Mohammad. Low-cost on-chip structures for combating die and IC recycling. In *Proceedings of the 51st Annual Design Automation Conference* (New York, NY, USA, 2014), DAC '14, ACM, pp. 87:1–87:6.

- [47] Hamalainen, P., Alho, T., Hannikainen, M., and Hamalainen, T. D. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In *9th EUROMICRO Conference on Digital System Design (DSD'06)* (2006), pp. 577–583.
- [48] Hayward, James A, and Meraglia, Janice. DNA marking and authentication: A unique, secure anti-counterfeiting program for the electronics industry. In *International Symposium on Microelectronics* (2011), vol. 2011, International Microelectronics Assembly and Packaging Society, pp. 000107–000112.
- [49] Holcomb, Daniel E., Burleson, Wayne P., and Fu, Kevin. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Transactions on Computers* 58, 9 (Sept. 2009), 1198–1210.
- [50] Huang, Wan-Chiech, Hsu, Chao-Ming, and Yang, Cheng-Fu. Recycling and refurbishing of epoxy packaging mold ports and plungers. *Inventions* 1, 2 (2016), 11.
- [51] Huda, Safeen, and Anderson, Jason. Towards PVT-tolerant glitch-free operation in FPGAs. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (New York, NY, USA, 2016), FPGA '16, ACM, pp. 90–99.
- [52] IHS Technology. Top 5 most counterfeited parts represent a \$169 billion potential challenge for global semiconductor market, 2012. Available: [http://www.isuppli.com/Semiconductor-Value-Chain/News/pages/Top-5-Most-Counterfeited-Parts-Represent-a-\\$169-Billion-Potential-Challenge-for-Global-Semiconductor-Market.aspx](http://www.isuppli.com/Semiconductor-Value-Chain/News/pages/Top-5-Most-Counterfeited-Parts-Represent-a-$169-Billion-Potential-Challenge-for-Global-Semiconductor-Market.aspx).
- [53] Ironwood Electronics. BGA socket, 2018. https://www.ironwoodelectronics.com/catalog/Content/Templates/PartGrids.cfm?StartRow=161&cPart=SG-BGA-6455&Grid=SG-BGA_TABLE-1mm.
- [54] Jarvinen, T., Salmela, P., Hamalainen, P., and Takala, J. Efficient byte permutation realizations for compact AES implementations. In *2005 13th European Signal Processing Conference* (Sept 2005), pp. 1–4.
- [55] Kae-Nune, N., and Pessegueir, S. Qualification and testing process to implement anti-counterfeiting technologies into IC packages. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2013), pp. 1131–1136.
- [56] Kaps, Jens-Peter, and Sunar, Berk. Energy comparison of AES and SHA-1 for ubiquitous computing. In *International Conference on Embedded and Ubiquitous Computing* (2006), Springer, pp. 372–381.
- [57] Karaklaji, D., Schmidt, J., and Verbaauwhede, I. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 12 (Dec 2013), 2295–2306.

- [58] Kerckhof, Stéphanie, Durvaux, François, Hocquet, Cédric, Bol, David, and Standaert, François-Xavier. *Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 390–407.
- [59] Kerry, C, and Gallagher, P. FIPS PUB 186-4: Digital Signature Standard (DSS). *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION*. National Institute of Standards und Technology (2013).
- [60] Keysight. B2901A Precision Source/Measure Unit. <https://www.keysight.com/en/pd-1983568-pn-B2901A/precision-source-measure-unit-1-ch-100-fa-210-v-3-a-dc-105-a-pulse?cc=US&lc=eng>.
- [61] Keysight. MSOX4154A Mixed Signal Oscilloscope. <https://www.keysight.com/en/pdx-x201943-pn-MSOX4154A/mixed-signal-oscilloscope-15-ghz-4-analog-plus-16-digital-channels?cc=US&lc=eng>.
- [62] Khovratovich, Dmitry, and Nikolić, Ivica. Rotational cryptanalysis of arx. In *Fast Software Encryption* (Berlin, Heidelberg, 2010), Seokhie Hong and Tetsu Iwata, Eds., Springer Berlin Heidelberg, pp. 333–346.
- [63] Kocher, Paul, Jaffe, Joshua, Jun, Benjamin, and Rohatgi, Pankaj. Introduction to differential power analysis. *Journal of Cryptographic Engineering* 1, 1 (2011), 5–27.
- [64] Kocher, Paul C., Jaffe, Joshua, and Jun, Benjamin. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology* (1999), CRYPTO '99, pp. 388–397.
- [65] Koziel, Eric, Thurmer, Kate, Milechin, Lauren, Grossmann, Peter, Vai, Michael, Khazan, Roger, Bergevin, Keith, and Comer, Philip. Side channel authenticity discriminant analysis for device class identification.
- [66] Lamoureux, J., Lemieux, G. G. F., and Wilton, S. J. E. Glitchless: Dynamic power minimization in FPGAs through edge alignment and glitch filtering. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16, 11 (Nov 2008), 1521–1534.
- [67] Leef, Serge. Supply Chain Hardware Integrity for Electronics Defense (SHIELD), 2018. Available: https://csrc.nist.gov/CSRC/media/Projects/cyber-supply-chain-risk-management/documents/SSCA/Winter_2018/TuePM2.1-SHIELD.pdf.
- [68] Leutenegger, S., Chli, M., and Siegwart, R. Y. BRISK: binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision* (Nov 2011), pp. 2548–2555.

- [69] Lim, Hyeonmin, Lee, Kyungsoo, Cho, Youngjin, and Chang, Naehyuck. Flip-flop insertion with shifted-phase clocks for FPGA power reduction. In *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* (Nov 2005), pp. 335–342.
- [70] Lin, Lang, and Burleson, W. Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* (May 2008), pp. 252–255.
- [71] Liu, D., Yu, C., Zhang, X., and Holcomb, D. Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2016), pp. 433–438.
- [72] Lowe, David G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (Nov 2004), 91–110.
- [73] Lukáš, Jan, Fridrich, Jessica, and Goljan, Miroslav. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security* 1, 2 (2006), 205–214.
- [74] Maes, Roel, and Verbauwhede, Ingrid. Physically unclonable functions: a study on the state of the art and future research directions. In *in Towards Hardware-Intrinsic Security, Security and Cryptology* (2010).
- [75] Maiti, Abhranil, Gunreddy, Vikash, and Schaumont, Patrick. A systematic method to evaluate and compare the performance of physical unclonable functions. cryptology eprint archive, report 2011/657, 2011.
- [76] Manncorp. SMT Pick and Place Machines, 2019. <https://www.manncorp.com/component-placement-and-handling>.
- [77] Massad, Mohamed El, Garg, Siddharth, and Tripunitara, Mahesh V. Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015* (2015).
- [78] Mathew, S., Satpathy, S., Suresh, V., Anders, M., Kaul, H., Agarwal, A., Hsu, S., Chen, G., and Krishnamurthy, R. 340 mV;1.1 V, 289 Gbps/W, 2090-Gate NanoAES Hardware Accelerator With Area-Optimized Encrypt/Decrypt GF(2⁴)² Polynomials in 22 nm Tri-Gate CMOS. *IEEE Journal of Solid-State Circuits* 50, 4 (April 2015), 1048–1058.
- [79] Michael L. Jones. DNA marking technology improves quality through fraud prevention, Sept 2016. Available: <http://www.dla.mil/AboutDLA/News/NewsArticleView/Article/958928/dna-marking-technology-improves-quality-through-fraud-prevention/>.

- [80] Microchip Technology Inc. 23A1024/23LC1024: 1Mbit SPI Serial SRAM with SDI and SQI Interface, 2015. <http://ww1.microchip.com/downloads/en/DeviceDoc/20005142C.pdf>.
- [81] Monteiro, J., Devadas, S., and Ghosh, A. Retiming sequential circuits for low power. In *Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on* (Nov 1993), pp. 398–402.
- [82] Muja, Marius, and Lowe, David G. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)* (2009), Alpesh Ranchordas and Helder Arajo, Eds., INSTICC Press, pp. 331–340.
- [83] Musoll, Enric, and Cortadella, Jordi. Low-power array multipliers with transition-retaining barriers. In *Power and Timing Modeling, Optimization and Simulation (PATMOS)* (Oct. 1995), pp. 227–238.
- [84] NASA JPL/OSMS Assurance Technology Program Office. Electric, Electronic and Electromechanical Parts Bulletin newsletter, 2011. available at https://nepp.nasa.gov/files/20647/2011%20EEE%20Parts%20Bulletin%20MayJune11%206_22_11.pdf.
- [85] National Research Council. *Counterfeit deterrent features for the next-generation currency design*, vol. 472. National Academies Press, 1993.
- [86] Nohl, Karsten, Evans, David, Starbug, Starbug, and Plötz, Henryk. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX security symposium* (2008), vol. 28.
- [87] Paar, Christof, and Pelzl, Jan. *Understanding Cryptography: A Textbook for Students and Practitioners*, 1st ed. Springer Publishing Company, Incorporated, 2009.
- [88] Pecht, M., and Tiku, S. Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. *IEEE Spectrum* 43, 5 (May 2006), 37–46.
- [89] Popp, Thomas, Kirschbaum, Mario, Zefferer, Thomas, and Mangard, Stefan. *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop*. 2007, ch. Evaluation of the Masked Logic Style MDPL on a Prototype Chip, pp. 81–94.
- [90] Popp, Thomas, and Mangard, Stefan. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005* (2005).
- [91] Pub, NIST FIPS. 197: Advanced encryption standard AES. *Federal Information Processing Standards Publication 197* (2001), 441–0311.

- [92] Rajendran, Jeyavijayan, Sam, Michael, Sinanoglu, Ozgur, and Karri, Ramesh. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (2013), CCS '13, pp. 709–720.
- [93] Ramesh, Chethan, Patil, Shivukumar B., Dhanuskodi, Siva Nishok, Provelengios, George, Pillement, Sébastien, Holcomb, Daniel, and Tessier, Russell. FPGA side channel attacks without physical access. In *26th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) 2018* (2018).
- [94] Report of the Committee on Armed Services United States Senate; 112th congress. INQUIRY INTO COUNTERFEIT ELECTRONIC PARTS IN THE DEPARTMENT OF DEFENSE SUPPLY CHAIN, 2012. Available: <https://www.armed-services.senate.gov/imo/media/doc/Counterfeit-Electronic-Parts.pdf>.
- [95] Roche, Thomas, Lomné, Victor, and Khalfallah, Karim. Combined Fault and Side-channel Attack on Protected Implementations of AES. In *Proceedings of the 10th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications* (Berlin, Heidelberg, 2011), CARDIS'11, Springer-Verlag, pp. 65–83.
- [96] Rublee, Ethan, Rabaud, Vincent, Konolige, Kurt, and Bradski, Gary. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision* (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 2564–2571.
- [97] SAE International. Counterfeit Electronic Parts; Avoidance, Detection, Mitigation, and Disposition, Users, 2012. Revised 2016-09-12.
- [98] Senate Armed Services Committee Hearing on Counterfeit Electronic Parts in the Defense Supply Chain. TESTIMONY OF RALPH L. DENINO Vice President Corporate Procurement L-3 Communications Corporation, Nov 2011. Available: <https://www.armed-services.senate.gov/imo/media/doc/DeNino%2011-08-11.pdf>.
- [99] Sharma, Ashlesh, Srinivasan, Vidyuth, Kanchan, Vishal, and Subramanian, Lakshminarayanan. The fake vs real goods problem: Microscopy and machine learning to the rescue. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2017), KDD '17, ACM, pp. 2011–2019.
- [100] Singh, A., Kar, M., Ko, J. H., and Mukhopadhyay, S. Exploring power attack protection of resource constrained encryption engines using integrated low-dropout regulators. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)* (July 2015).

- [101] Skorobogatov, Sergei, and Woods, Christopher. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems* (2012), CHES'12, pp. 23–40.
- [102] Skudlarek, J. P., Katsioulas, T., and Chen, M. A platform solution for secure supply-chain and chip life-cycle management. *Computer* 49, 8 (Aug 2016), 28–34.
- [103] SparkFun Electronics Blog. Fake ICs Identified, July 2010. Available: <https://www.sparkfun.com/news/395>.
- [104] Spreitzer, Raphael, Moonsamy, Veelasha, Korak, Thomas, and Mangard, Stefan. Systematic classification of side-channel attacks: A case study for mobile devices. *IEEE Communications Surveys and Tutorials* 20, 1 (2018), 465–488.
- [105] Tiri, K., and Verbauwhede, I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition* (2004).
- [106] Tiri, K., and Verbauwhede, I. A digital design flow for secure integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 7 (July 2006), 1197–1208.
- [107] Tiri, Kris, et al. Prototype IC with WDDL and differential routing–DPA resistance assessment. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2005).
- [108] Tokunaga, C., and Blaauw, D. Secure AES engine with a local switched-capacitor current equalizer. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers* (Feb 2009).
- [109] Tong, KW, Kwong, CK, and Ip, KW. Optimization of process conditions for the transfer molding of electronic packages. *Journal of Materials Processing Technology* 138, 1 (2003), 361–365.
- [110] Tu-Hsiung Tsai, Hung-Ming Chen, Hung-Chun Li, Shi-Hao Chen. An efficient RDL routing for flip-chip designs., 2013. <https://www.edn.com/design/systems-design/4419930/An-efficient-RDL-routing-for-flip-chip-designs>.
- [111] Tummala, Rao R. Fundamentals of microsystems packaging.
- [112] Tuyls, Pim, Schrijen, Geert-Jan, Škorić, Boris, van Geloven, Jan, Verhaegh, Nynke, and Wolters, Rob. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems - CHES* (Berlin, Heidelberg, 2006), Springer Berlin Heidelberg, pp. 369–383.

- [113] ViTiny USA. ViTiny UM12 Long Working Distance 5MP USB Digital Microscope, 2018. <http://www.vitiny-usa.com/vitiny-um12.html>.
- [114] Wilton, Steven J. E., Ang, Su-Shin, and Luk, Wayne. *The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 719–728.
- [115] Yang, K., Blaauw, D., and Sylvester, D. Hardware Designs for Security in Ultra-Low-Power IoT Systems: An Overview and Survey. *IEEE Micro* 37, 6 (November 2017), 72–89.
- [116] Zhao, W., Ha, Y., and Alioto, M. AES architectures for minimum-energy operation and silicon demonstration in 65nm with lowest energy per encryption. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2015), pp. 2349–2352.
- [117] Zheng, Yu, Basak, Abhishek, and Bhunia, Swarup. CACI: Dynamic current analysis towards robust recycled chip identification. In *Proceedings of the 51st Annual Design Automation Conference* (New York, NY, USA, 2014), DAC '14, ACM, pp. 88:1–88:6.
- [118] Zygo. Nexview 3D Optical Surface Profiler. https://www.zygo.com/?/met/profilers/nexview/&utm_source=zygo&utm_medium=QualityMag&utm_content=NexviewPage&utm_campaign=PrintAd.