
Contents

Foreword	xxvii
Preface	xix
List of Figures	xxiii
List of Tables	xxxix
I Background	1
1 Mathematical Background	3
1.1 Introduction	3
1.2 Modular Arithmetic	4
1.3 Groups, Rings, and Fields	6
1.4 Greatest Common Divisors and Multiplicative Inverse	8
1.4.1 Euclidean Algorithm	9
1.4.2 Extended Euclidean Algorithm	10
1.4.3 Chinese Remainder Theorem	11
1.5 Subgroups, Subrings, and Extensions	13
1.6 Groups, Rings, and Field Isomorphisms	14
1.7 Polynomials and Fields	16
1.8 Construction of Galois Field	17
1.9 Extensions of Fields	18
1.10 Cyclic Groups of Group Elements	19
1.11 Efficient Galois Fields	21
1.11.1 Binary Finite Fields	21
1.12 Mapping between Binary and Composite Fields	24
1.12.1 Constructing Isomorphisms between Composite Fields	25
1.12.1.1 Mapping from $GF(2^k)$ to $GF(2^n)^m$, where $k = nm$	25
1.12.1.2 An Efficient Conversion Algorithm	26
1.13 Conclusions	28
2 Overview of Modern Cryptography	29
2.1 Introduction	30
2.2 Cryptography: Some Technical Details	31
2.3 Block Ciphers	36
2.3.1 Inner Structures of a Block Cipher	38
2.3.2 The Advanced Encryption Standard	39
2.3.3 The AES Round Transformations	40
2.3.3.1 SubBytes	41
2.3.3.2 ShiftRows	42
2.3.3.3 MixColumns	42

2.3.3.4	AddRoundKey	43
2.3.4	Key-Scheduling in AES	43
2.4	Rijndael in Composite Field	44
2.4.1	Expressing an Element of $GF(2^8)$ in Subfield	44
2.4.2	Inversion of an Element in Composite Field	45
2.4.3	The Round of AES in Composite Fields	46
2.5	Elliptic Curves	48
2.5.1	Simplification of the Weierstraß Equation	50
2.5.2	Singularity of Curves	51
2.5.3	The Abelian Group and the Group Laws	51
2.5.4	Elliptic Curves with Characteristic 2	52
2.5.5	Projective Coordinate Representation	57
2.6	Scalar Multiplications: LSB First and MSB First Approaches	58
2.7	Montgomery's Algorithm for Scalar Multiplication	59
2.7.1	Montgomery's Ladder	60
2.7.2	Faster Multiplication on EC without Pre-computations	60
2.7.3	Using Projective co-ordinates to Reduce the Number of Inversions	62
2.8	Conclusions	62

Modern Hardware Design Practices 65

3.1	Introduction	65
3.1.1	FPGA Architecture	66
3.1.2	The FPGA Design Flow	68
3.2	Mapping an Algorithm to Hardware: Components of a Hardware Architecture	72
3.3	Case study: Binary gcd Processor	73
3.4	Enhancing the Performance of a Hardware Design	75
3.5	Modelling of the Computational Elements of the gcd Processor	78
3.5.1	Modeling of an Adder	78
3.5.2	Modeling of a Multiplexer	79
3.5.3	Total LUT Estimate of the gcd Processor	80
3.5.4	Delay Estimate of the gcd Processor	80
3.6	Experimental Results	80
3.6.1	LUT utilization of the gcd processor	81
3.6.2	Delay Estimates for the gcd Processor	81
3.7	Conclusions	81

Hardware Design of Cryptographic Algorithms 83

Hardware Design of the Advanced Encryption Standard (AES) 85

4.1	Introduction	85
4.2	Algorithmic and Architectural Optimizations for AES Design	86
4.3	Circuit for the AES S-Box	86
4.3.1	Subfield Mappings	87
4.3.2	Circuit Optimizations in the Polynomial Basis	87
4.3.3	Circuit Optimizations in the Normal Basis	91
4.3.4	Most Compact Realization: Polynomial vs. Normal basis	94
4.3.5	Common Subexpressions	96
4.3.6	Merging of Basis Change Matrix and Affine Mapping	97
4.3.7	Merged S-Box and Inverse S-Box	97
4.4	Implementation of the MixColumns Transformation	98
4.4.1	The AES S-Box and MixColumns	99

4.4.2	Implementing the Inverse MixColumns	100
4.5	An Example Reconfigurable Design for the Rijndael Cryptosystem	101
4.5.1	Overview of the Design	101
4.5.1.1	Encryption Unit	102
4.5.1.2	Pipelining and Serializing in FPGAs	102
4.5.1.3	Back to the Design	103
4.5.2	Key-Schedule Optimization	106
4.5.3	The Control Unit	107
4.6	Experimental Results	109
4.7	Single Chip Encryptor/Decryptor	110
4.8	Conclusions	111

5 Efficient Design of Finite Field Arithmetic on FPGAs 115

5.1	Introduction	116
5.2	Finite Field Multiplier	116
5.3	Finite Field Multipliers for High Performance Applications	117
5.4	Karatsuba Multiplication	118
5.5	Karatsuba Multipliers for Elliptic Curves	118
5.6	Designing for the FPGA Architecture	119
5.7	Analyzing Karatsuba Multipliers on FPGA Platforms	120
5.7.1	The Hybrid Karatsuba Multiplier	123
5.8	Performance Evaluation	126
5.9	High-Performance Finite Field Inversion Architecture for FPGAs	126
5.10	Itoh-Tsujii Inversion Algorithm	127
5.11	The Quad ITA Algorithm	128
5.11.1	Clock Cycles for the ITA	129
5.11.2	The Quad ITA Generalization	130
5.11.3	Hardware Architecture of Quad ITA	133
5.11.3.1	Addition Chain Selection Criteria	136
5.11.3.2	Design of the Quadblock	136
5.12	Experimental Results	137
5.13	Generalization of the ITA for 2^n Circuit	138
5.14	Hardware Architecture for 2^n Circuit Based ITA	140
5.15	Area and Delay Estimations for the 2^n ITA	141
5.15.1	Area and Delay of a x Variable Boolean Function	141
5.15.2	Area and LUT Delay Estimates for the Field Multiplier	142
5.15.3	Area and LUT Delay Estimates for the Reduction Circuit	145
5.15.4	Area and LUT Delay Estimates of a 2^n Circuit	146
5.15.5	Area and LUT Delay Estimates of a Multiplexer	146
5.15.6	Area and LUT Delay Estimates for the Powerblock	146
5.15.7	Area Estimate for the Entire ITA Architecture	147
5.15.8	LUT Delay Estimate for the Entire ITA Architecture	147
5.16	Obtaining the Optimal Performing ITA Architecture	148
5.17	Validation of Theoretical Estimations	149
5.18	Conclusions	151

6 High-Speed Implementation of Elliptic Curve Scalar Multiplication on FPGAs 153

6.1	Introduction	153
6.2	The Elliptic Curve Cryptoprocessor	155
6.2.1	Register Bank	155

8.8.5	Other Techniques	254
8.9	Invariance based DFA Countermeasures	254
8.9.1	An Infective Countermeasure Scheme	254
8.9.2	Infection Countermeasure	255
8.9.3	Attacks on the Infection Countermeasure	256
8.9.3.1	Further Loop Holes in the Countermeasure: Attacking the Infection Technique	257
8.9.4	Infection Caused by Compulsory Dummy Round	257
8.9.4.1	Attacking the Top Row	258
8.9.5	Piret & Quisquater's Attack on the Countermeasure	259
8.10	Improved Countermeasure	262
8.11	Conclusions	264
Cache Attacks on Ciphers		265
9.1	Memory Hierarchy and Cache Memory	266
9.1.1	Organization of Cache Memory	267
9.1.2	Improving Cache Performance for Superscalar Processors	269
9.2	Timing Attacks Due to CPU Architecture	269
9.2.1	Attacks on Block Ciphers using Data Cache Memories	269
9.2.1.1	Table-based Implementation of the AES	269
9.2.1.2	Software Implementations of AES	271
9.3	Trace-Driven Cache Attacks	272
9.3.1	A Theoretical Cache Trace Attack on DES	273
9.3.2	Trace-Driven Attacks on AES	273
9.3.3	Trace-Driven Attacks on the Last Round of AES	274
9.3.4	Trace-Driven Attack Based on Induced Cache Miss	275
9.4	Access-Driven Cache Attacks	275
9.4.1	Access-Driven Attacks on Block Ciphers	276
9.4.2	Second Round Access-Driven Attack on AES	277
9.4.3	A Last Round Access Driven Attack on AES	278
9.4.4	Asynchronous Access-Driven Attacks	278
9.4.5	Fine-Grained Access-Driven Attacks on AES	279
9.5	Time-Driven Cache Attacks	280
9.5.1	Remote Timing Attacks	280
9.5.2	Distinguishing between a Hit and Miss with Time	281
9.5.3	Second Round Time-Driven Attack on AES	282
9.5.4	Theoretical Analysis of Time-Driven Attacks	283
9.5.5	Profiled Time-Driven Attack on AES	283
9.6	Countermeasures for Timing Attacks	286
9.6.1	Application-Layer Countermeasures	287
9.6.2	Countermeasures Applied in the Hardware	288
9.7	Conclusions	291
Power Analysis of Cipher Implementations		293
10.1	Power Attack Setup and Power Traces	294
10.1.1	Power Traces of an AES hardware	295
10.1.2	Quality of Power Measurements	297
10.2	Power Models	298
10.2.1	Hamming Weight Power Model	298
10.2.2	Hamming Distance Power Model	298
10.2.3	Why Do Gates Leak?	300

10.3	Differential Power Analysis Using Difference of Mean	301
10.3.1	Computing DoM for Simulated Power Traces on AES	302
10.3.2	Simulation of the Power Traces and Computing the DoM	302
10.3.3	DOM and the Hamming Distance vs. Hamming Weight Model	303
10.4	PKDPA: An Improvement of the DoM technique	304
10.4.1	Application of PKDPA to AES	304
10.4.2	Choosing the Window-Size	306
10.5	Correlation Power Attack	309
10.5.1	Computing Correlation Coefficient for Simulated Power Traces for AES	310
10.5.2	Computation of the Correlation Matrix	311
10.5.3	Experimental Results on Simulated Traces	312
10.6	Metrics to Evaluate a Side Channel Analysis	312
10.6.1	Success Rate of a Side Channel Adversary	312
10.6.2	Guessing Entropy of an Adversary	313
10.7	CPA on Real Power Traces of AES-128	315
10.7.1	SNR of a Power Trace	315
10.7.2	DPA of an Iterative Architecture	317
10.7.3	DPA on a Serialized Architecture of AES	318
10.7.4	Shuffling Architecture	318
10.8	Popular Countermeasures against Power Analysis: Masking	321
10.8.1	Masking at the Algorithmic Level	322
10.8.1.1	Masking the AES S-Box	322
10.8.1.2	Security of the Masking Scheme	324
10.8.1.3	Circuit Complexity and Reuse of Masks	325
10.8.2	Masking at the Gate Level	327
10.8.3	The Masked AND Gate and Vulnerabilities due to Glitches	328
10.9	Conclusions	330
11 Testability of Cryptographic Hardware		333
11.1	Introduction	333
11.2	Scan Chain Based Attacks on Cryptographic Implementations	334
11.2.1	Scan Chain Based Attack on Stream Ciphers	335
11.3	Scan Attack on Trivium	336
11.3.1	Brief Description of Trivium	336
11.3.1.1	Key and IV Setup	336
11.3.2	KeyStream Generation	337
11.3.3	The Attack	337
11.3.3.1	Objective of the Attacker	337
11.3.3.2	Attack on Trivium	337
11.3.3.3	Ascertaining Bit Correspondence	338
11.3.4	Deciphering the Cryptogram	339
11.4	Testability of Cryptographic Designs	341
11.4.0.1	Reset Attack on Flipped-Scan Mechanism	342

11.5 Conclusions	345
7 Hardware Intellectual Property Protection	347
Hardware Intellectual Property Protection through Obfuscation	349
12.1 Introduction	350
12.2 Related Work	352
12.3 Functional Obfuscation through State Transition Graph Modification	353
12.3.1 Goals of the Obfuscation Technique	353
12.3.2 Hardware IP Piracy: Adversary's Perspective	353
12.3.2.1 Modification Scheme Employing Input Logic-cone Expansion	354
12.3.3 Obfuscation Metric	355
12.3.4 System-level Obfuscation	356
12.3.4.1 State Transition Function Modification	356
12.3.5 Embedding Authentication Features	357
12.3.6 Choice of Optimal Set of Nodes for Modification	358
12.3.7 The <i>HARPOON</i> Design Methodology	359
12.4 Extension of STG Modification for RTL Designs	362
12.5 Obfuscation through Control and Dataflow Graph (CDFG) Modification	364
12.5.1 CDFG Obfuscation Methodology	364
12.5.1.1 Embedding Authentication Features	367
12.5.2 Comparison between the Two Approaches	367
12.5.3 Obfuscation-based Secure SoC Design Flow	368
12.6 Measure of Obfuscation Level	368
12.6.1 Manual Attacks by Visual Inspection	368
12.6.2 Simulation-based Attacks	369
12.6.3 Structural Analysis-based Attack	369
12.6.3.1 Structural Analysis against STG Modification	369
12.6.3.2 Structural Analysis against CDFG Modification-based Approach	371
12.6.4 Quantitative Comparison	371
12.7 Results	372
12.7.1 Design Flow Automation	372
12.7.2 Implementation Results	374
12.7.2.1 Implementation Setup	374
12.7.2.2 Obfuscation Efficiency and Overheads	374
12.7.3 Effect of Key Length	375
12.7.3.1 Support for Multiple-length Keys	375
12.7.3.2 Effect of Increasing Key Length	376
12.8 Discussions	377
12.8.1 Using Unreachable States during Initialization	377
12.9 Conclusions	378
7 Hardware Trojans	379
3 Overview of Hardware Trojans	381
13.1 Introduction	382
13.2 Trojan Taxonomy and Examples	383
13.3 Multi-level Attacks	386

13.3.1 Difficulty of Recovering Encryption Key from Modified Faulty Cipher-text	389
13.3.2 Effectiveness of Multi-level Attacks	389
13.3.3 Results: Multi-level Attacks	390
13.3.4 Extension of the Multi-level Attack Model	391
13.3.5 Prevention Techniques against Multi-level Attacks	391
13.4 Effect of Hardware Trojan on Circuit Reliability	392
13.4.1 Hardware Trojan and Reliability: Analysis and Results	394
13.4.1.1 Combinational Trojans	394
13.4.2 Synchronous Counter-based Sequential Trojan	395
13.4.3 Asynchronous Counter-based Sequential Trojan	397
13.5 Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream	398
13.5.1 Xilinx Configuration Bitstream Structure	400
13.5.2 Xilinx Virtex-II FPGA Building Blocks	400
13.5.3 Xilinx Configuration Bitstream File Organisation	400
13.5.3.1 Configuration Memory Addressing	400
13.5.4 Configuration Process	401
13.5.5 Cyclic Redundancy Check (CRC)	401
13.5.6 Disabling CRC Check	401
13.5.7 Bitstream Modification Technique	401
13.5.8 Demonstration of the Bitstream Modification-based Attack	404
13.5.9 Prevention/Detection Techniques against the Proposed Attack	406
13.5.9.1 Grounding Unused Pins in a FPGA	406
13.5.9.2 Online Temperature Monitoring	406
13.5.9.3 Filling-up Unused Resources of the FPGA	406
13.5.10 Logic Distribution to Balance Power Consumption from the I/O Power Pins	406
13.5.10.1 Dedicated Hardware Logic to Check CRC Status	406
13.5.10.2 Scrambling and De-scrambling the Bitstream File	406
13.6 Conclusion	407
14 Logic Testing based Hardware Trojan Detection	409
14.1 Introduction	409
14.2 Statistical Approach for Trojan Detection	410
14.2.1 Mathematical Analysis	411
14.2.2 Test Generation	412
14.2.3 Coverage Estimation	413
14.2.4 Choice of Trojan Sample Size	413
14.2.5 Choice of N	413
14.2.6 Improving Trojan Detection Coverage	413
14.3 Results	415
14.3.1 Simulation Setup	415
14.3.2 Comparison with Random and ATPG Patterns	416
14.3.3 Effect of Number of Trigger Points (q)	417
14.3.4 Effect of Trigger Threshold (θ)	418
14.3.5 Sequential Trojan Detection	418
14.3.6 Application to Side-channel Analysis	420
14.4 Conclusions	420

15 Side-channel Analysis Techniques for Hardware Trojans Detection	421
15.1 Introduction	422
15.2 Motivation for the Proposed Approaches	423
15.3 Multiple-parameter Analysis based Trojan Detection	424
15.3.1 Theoretical Analysis	425
15.3.2 Improving Detection Sensitivity	428
15.3.2.1 Test Vector Selection	428
15.3.2.2 Power Gating and Operand Isolation	430
15.3.2.3 Use of Other Side-channel Parameters	431
15.3.2.4 Test Conditions	432
15.4 Results	433
15.4.1 Simulation Verification	433
15.4.1.1 Test Setup	433
15.4.1.2 Results	434
15.4.2 Hardware Validation	436
15.4.2.1 Test Setup	436
15.4.2.2 Results	438
15.5 Integration with Logic-testing Approach	439
15.5.1 Motivation of Self-Referencing Approach	441
15.5.2 Analysis of Process Variations on Self-Referencing	443
15.5.3 Self-Referencing Methodology	444
15.5.3.1 Functional Decomposition	445
15.5.3.2 Statistical Test Vector Generation	446
15.5.3.3 Side-channel Analysis Using Self-Referencing	447
15.5.3.4 Decision Making Process	448
15.5.4 Self-referencing Simulation Results	449
15.5.5 Experimental Results	450
15.5.6 Conclusions	451
6 Design Techniques for Hardware Trojan Threat Mitigation	453
16.1 Introduction	453
16.2 Obfuscation-based Trojan Detection/Protection	454
16.2.1 Methodology of Circuit Obfuscation-based Trojan Detection / Protection	454
16.2.2 Effect of Obfuscation on Trojan Insertion	456
16.2.3 Effect of Obfuscation on Trojan Potency	457
16.2.4 Effect of Obfuscation on Trojan Detectability	458
16.2.5 Effect of Obfuscation on Circuit Structure	459
16.2.6 Determination of Unreachable States	460
16.2.7 Test Generation for Trojan Detection	460
16.2.8 Determination of Effectiveness	461
16.3 Integrated Framework for Obfuscation	462
16.4 Results	464
16.4.1 Discussions	466
16.4.2 Application to Third-party IP Modules and SoCs	466
16.4.3 Protection against Malicious CAD Tools	466
16.4.4 Improving Level of Protection and Design Overhead	469
16.4.5 Protection against Information Leakage Trojans	469
16.5 A FPGA-based Design Technique for Trojan Isolation	470
16.6 A Design Infrastructure Approach to Prevent Circuit Malfunction	471

VI Physically Unclonable Functions	473
17 Physically Unclonable Functions: a Root-of-Trust for Hardware Security	475
17.1 Introduction	476
17.2 Physically Unclonable Function	476
17.3 Classification of PUFs	478
17.3.1 Classification based on Technology	478
17.3.2 Classification based on Response Collection Mechanism	479
17.3.3 Classification based on Difficulty of Modeling	480
17.3.4 Classification based on Reconfiguration Technique	480
17.4 Realization of Silicon PUFs	481
17.4.1 Delay-based PUF	481
17.4.1.1 Ring Oscillator PUF	481
17.4.1.2 Arbiter PUF	482
17.4.1.3 Bistable Ring PUF	483
17.4.1.4 Loop PUF	483
17.4.2 Memory-based PUF	484
17.4.2.1 SRAM PUF	484
17.4.2.2 Butterfly PUF	485
17.4.2.3 Latch PUF	485
17.4.2.4 Flip-flop PUF	485
17.5 PUF Performance Metrics for Quality Evaluation	486
17.5.1 Uniqueness	486
17.5.2 Reliability	487
17.5.3 Uniformity	488
17.5.4 Bit-aliasing	488
17.5.5 Bit-dependency	488
17.6 Secure PUF: What Makes a PUF Secure?	488
17.7 Applications of PUF as a Root-of-Trust	489
17.7.1 Low-cost Authentication	489
17.7.2 Cryptographic Key Generation	489
17.7.3 IP protection of FPGA Hardware Designs	491
17.7.4 Key Zeroization	491
17.8 Attacks Model: How PUF Security Could Be Compromised?	491
17.8.1 Fault Attacks on PUFs and Fuzzy Extractors	492
17.8.2 Side-channel Attacks on PUFs	492
17.8.3 Modeling Attacks on Delay-based PUFs	492
17.9 Looking Forward: What Lies Ahead for PUFs?	493
18 Genetic Programming-based Model-building Attack on PUFs	495
18.1 Introduction	495
18.2 Background: Genetic Programming and RO-PUFs	497
18.2.1 Genetic Programming	497
18.2.2 Ring Oscillator PUF (RO-PUF)	498
18.3 Methodology	499
18.3.1 Reproduction Schemes	500
18.3.1.1 Elitist Model	500
18.3.1.2 Tournament Selection Model	500
18.4 Results	500
18.4.1 Experimental Setup	500
18.4.2 Conclusions	502