DIPLOMARBEIT

Seitenkanalangriffe auf Verschlüsselungsalgorithmen und deren Verhinderung

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

em. o. Univ. Prof. Dr. Dietmar Dietrich
Dr. Florian Schupfer

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Lukas Wimmer
Matr.Nr. e0926090
Oberes Dorf, 2620 Hafning

12. Januar 2015 _____

Kurzfassung

Neben mathematischen Angriffsmethoden auf Verschlüsselungen existieren seit einigen Jahren auch alternative Angriffsmöglichkeiten über sogenannte Seitenkanäle, welche die aktuelle elektrische Verlustleistung eines Geräts oder die Dauer von Berechnungen benutzen. Damit können Rückschlüsse auf interne Schalt- und Rechenvorgänge gezogen und der geheime Schlüssel extrahiert werden. Die asymmetrische Verschlüsselungsmethode RSA lässt sich mit Hilfe von verschiedenen Algorithmen in einem „Field Programmable Gate Array“ umsetzen, wodurch eine unterschiedliche Interpretation des Seitenkanals nötig ist. Diese Diplomarbeit beschäftigt sich daher mit Methoden, um diese Algorithmen in einer Blackbox zu identifizieren.
Aus den Implementierungen der Algorithmen, die auf dem chinesischen Restsatz, dem binären modularen Potenzieren, der Montgomery- oder der Blakley-Multiplikation basieren, wurden im Zuge dieser Arbeit die geheimen Schlüssel erfolgreich extrahiert. Die Ergebnisse dazu fließen in die Erarbeitung von möglichst generische Detektionsmethoden ein. Viele mathematisch sichere Implementierungen von Verschlüsselungen achten nicht auf die Verwundbarkeit im Seitenkanal. Auch wurden einige Schutzmaßnahmen aus der Vergangenheit durch verbesserte Angriffs- und Auswertungsmöglichkeiten umgangen. Daher stehen neben der Algorithmenidentifikation auch Gegenmaßnahmen im Fokus, die diese erschweren oder verhindern können.

Abstract

Aside from the common mathematical approaches for breaking encryption methods a growing number of alternative attacks exist, which are referred to as side channel attacks. Those approaches are using alternative ways like measuring power consumption or computing time of a device. Using this information internal operations can be revealed and secret keys extracted. The asymmetric encryption RSA can be implemented in a field programmable gate array by using different kinds of algorithms, thus requiring different interpretations of the side-channel. This thesis is about methods to identify those algorithms in a blackbox.
The analysed implementations consist of the Chinese remainder theorem, the binary exponentiation algorithm, the Montgomery- or the Blakley-Multiplication and were successfully attacked in this work to reveal the hidden key of the implementation. The results were used to develop generic detection methods. Many encryption implementations, which are proven as mathematically secure, do not worry about side-channel leakage. Also a lot of countermeasures were cracked through enhanced attacks and analyses in the past. Therefore, besides the prevention of the algorithm identification, focus is put on countermeasures, which complicate or even block those previously mentioned attacks.

Danksagung

An dieser Stelle möchte ich mich bei meiner Familie und meiner Freundin bedanken, die mich während der Zeit meines Studiums stets unterstützt haben. Sie waren es, die mir beim Verfassen der Diplomarbeit, besonders in der Endphase mit Rat und Tat zur Seite standen, indem sie meine Arbeit Korrektur gelesen haben und einige Ratschläge und Anmerkungen für mich parat hatten. Sie schafften es, mich auch in den schwierigen Phasen des Schreibens stets neu zu motivieren und brachten mich außerdem dazu, die letzten Monate nicht nur am Schreibtisch zu sitzen, sondern auch die nötigen Pausen nicht zu vergessen.
Ein weiterer Dank gilt meinem Betreuer Dr. Florian Schupfer, der durch seine umfangreiche Unterstützung dazu beigetragen hat, dass sich diese Arbeit in die richtige Richtung entwickelte.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 2
- 1.2 Problemstellung . . . 4
- 1.3 Aufgabenstellung und Methodik . . . 4

2 Verwandte Arbeiten **6**
- 2.1 Erste Seitenkanalangriffe . . . 6
- 2.2 Seitenkanalangriffe auf Field Programmable Gate Arrays . . . 7
- 2.3 Detektion von Algorithmen . . . 9
- 2.4 Gegenmaßnahmen . . . 9

3 Verschlüsselungsmethoden **11**
- 3.1 Mathematische Grundlagen . . . 11
- 3.2 RSA – Algorithmus . . . 12
  - 3.2.1 Chinesischer Restsatz . . . 14
  - 3.2.2 Modulares Potenzieren . . . 15
  - 3.2.3 Modulare Multiplikation . . . 18
- 3.3 Anwendungen . . . 20
  - 3.3.1 Digitale Signatur und Zertifikate . . . 21
  - 3.3.2 Verschlüsselungen in verteilten Netzen . . . 22
- 3.4 RSA-ähnliche Verschlüsselungsverfahren . . . 22
  - 3.4.1 Diffie-Hellman Schlüsselaustausch . . . 22
  - 3.4.2 Elgamal Verschlüsselung . . . 23

4 Seitenkanalangriffsvektoren **24**
- 4.1 Simple Power Analysis . . . 25
- 4.2 Differential Power Analysis . . . 26
- 4.3 Rechenzeitanalyse . . . 29
- 4.4 Electro-Magnetic Analysis . . . 30
- 4.5 Fault Analysis . . . 30
- 4.6 Gegenmaßnahmen . . . 31

5 Methoden zur Detektion von ausgewählten Algorithmen **34**
- 5.1 Angriffsmethodik . . . 35
- 5.2 Untersuchung und Auswirkung der Bitlänge . . . 35

5.3 Variation der Eingangsdaten . . . 36
5.4 Analyse spezieller verschlüsselter Nachrichten . . . 38
5.5 Abschnittsanalyse . . . 39
5.6 Abgleich mit einem Simulationsmodell . . . 40
5.7 Ungeeignete Angriffsvektoren . . . 40
5.8 Analyse der Schutzmaßnahmen gegen Detektionsmethoden . . . 41

6 Seitenkanalangriffe auf ausgewählte Implementierungen 46
6.1 „Quadrieren und Multiplizieren“-„Blakley“-Algorithmus . . . 47
6.2 „Chinesischer Restsatz“-„Blakley“-Algorithmus . . . 53
6.3 „Quadrieren und Multiplizieren“-„Montgomery“-Algorithmus . . . 56
6.4 „Chinesischer Restsatz“-„Montgomery“-Algorithmus . . . 61
6.5 Verifikation der Ergebnisse . . . 63

7 Analyse des Messumfeldes und Charakteristiken schaltender Logik 65
7.1 Messumfeld . . . 65
7.2 Spartan-6 Entwicklungsboard . . . 66
7.3 Erkenntnisse aus Voruntersuchungen . . . 68

8 Zusammenfassung 72
8.1 Ergebnisse und Erkenntnisse . . . 72
8.2 Ausblick . . . 75

Anhang 76
A.1 Kryptographie auf elliptischen Kurven . . . 76
B.2 Digilent Adept System . . . 79
C.3 Entfernte Stützkapazitäten . . . 80

Wissenschaftliche Literatur 81

Internet Referenzen 85

Abkürzungen

ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal-Oxide-Semiconductor
CPA	Correlation Power Analysis
CPS	Cyber-Physical System
CRT	Chinese Remainder Theorem
DLP	Diskretes Logarithmisches Problem
DPA	Differential Power Analysis
ECC	Elliptic Curve Cryptography
EMA	Electro-Magnetic Analysis
FA	Fault Analysis
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GPIB	General Purpose Interface Bus
HO-DPA	High-Order Differential Power Analysis
HTTPS	HyperText Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
NMOS	N-Channel Metal-Oxide-Semiconductor
OSI	Open Systems Interconnection
PMOS	P-Channel Metal-Oxide-Semiconductor
RSA	Rivest Shamir und Adleman
SNR	Signal-to-Noise Ratio
SoC	System on Chip
SPA	Simple Power Analysis
SRAM	Static Random-Access Memory
SSH	Secure Shell
SSL	Secure Socket Layer
TA	Template-based Analysis
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

Mathematische Symbole

Diese Tabelle enthält alle in dieser Diplomarbeit verwendeten mathematischen Symbole.

Symbol	Definition
$ggT(x, y)$	Größter gemeinsamer Teiler der beiden Zahlen x und y
$\varphi(n)$	Eulersche φ-Funktion mit dem Parameter n
$\circ$	Multiplikationszeichen im Montgomery-Raum
$\overline{x}$	Repräsentiert die Zahl x im Montgomery-Raum
$\#$	Mengenoperator

Liste der Algorithmen

3.1 Binäres modulares Potenzieren von links nach rechts 16
3.2 Binäres modulares Potenzieren von rechts nach links 16
3.3 m-ary Methode 17
3.4 Blakley-Multiplikation 19
3.5 Potenzieren im Montgomery-Raum 20
4.1 Montgomery Power Ladder 32

1 Einleitung

Kryptographie kommt in unserem Alltag immer häufiger zum Einsatz und das oftmals unbemerkt. Sie verhindert den einfachen Zugriff Dritter auf übertragene Informationen und die Manipulation dieser. Sei es, wenn man eine Suchabfrage im Internet startet oder an einem Bankomat Bargeld mit einer Smartcard behebt. Durch die technische Weiterentwicklung werden immer mehr Geräte und Applikationen über digitale Netzwerke miteinander gekoppelt. Viele neue Anwendungen, wie zum Beispiel der intelligente Stromzähler oder andere teure Prozessanlagen in großen Firmen, gehören dabei zu einer unverzichtbaren Infrastruktur, deren Kommunikation besonders schützenswert ist. Aus diesem Grund setzt man auf bewährte Verschlüsselungsstandards, die vor solchen Angriffen schützen. Dem Problem des Verteilens von geheimen Schlüsselinformationen kann durch asymmetrische Verschlüsselungen und Schlüsselaustauschmethoden begegnet werden.

Dazu zählen vor allem die Kryptosysteme RSA (Rivest, Shamir und Adleman), ECC („Elliptic Curve Cryptography“), Elgamal und Deffie-Hellman. Diese Arbeit legt einen besonderen Fokus auf das asymmetrische Verschlüsselungssystem RSA, da es in vielen Protokollen verwendet wird.

Diese aufgezählten Systeme werden von Kryptographen als sicher eingestuft, da noch keine effektive Methode existiert, welche das diskrete logarithmische Problem lösen oder große Zahlen faktorisieren kann. Zusätzlich zu diesen mathematischen Angriffsmöglichkeiten sind seit einigen Jahren auch alternative Angriffsmethoden im Einsatz, die darauf basieren, dass sich diese mathematischen Operationen, die für eine Verschlüsselung notwendig sind, auf einem realen System wiederfinden. Die dafür notwendigen Informationen, wie die momentane Verlustleistung und die elektromagnetische Abstrahlung eines Geräts oder die Ausführungszeit eines Verschlüsselungsvorgangs, lassen auf interne Vorgänge in einem Device rückschließen.

Der Erfolg eines sogenannten Seitenkanalangriffs [KSWH98] hängt von vielen Umgebungsparametern ab. Neben der Messbarkeit und Zugänglichkeit der Informationen im verwendeten Kanal, spielen der Typus des Geräts und die damit verbundene Implementierung eine bedeutsame Rolle in der Analyse. Vergleicht man den Seitenkanal der elektrischen Verlustleistung eines „Field Programmable Gate Arrays“ (FPGAs) mit einer herkömmlichen Prozessorarchitektur, gibt es in der Interpretation merkbare Unterschiede. Ein FPGA kann in einem Taktschritt durch seine Parallelität gleichzeitig Operationen ausführen. Im Unterschied dazu wird in einer „Von-Neumann-Architektur“ zwischen dem Laden der Register und der Ausführungsphase von mathematischen Operationen unterschieden.

1.1 Motivation

Laut [SSK10] hat die Entwicklung in den letzten Jahren gezeigt, dass „System on Chips“ (SoCs) - ein monolithisches System auf einem Chip, das verschiedene Schaltungsteile integriert hat - immer öfters auf Strukturen mit FPGAs setzen. Neben der hohen nutzbaren Bandbreite zwischen Prozessor und FPGA hat dies den Vorteil, dass dadurch ein geringerer Leistungsverbrauch und eine höhere Flexibilität geschaffen wird. Weiters können durch eine kurze Produkteinführungszeit („Time to Market“) und eine erleichterte Prototypenherstellung Entwicklungskosten gespart werden, was zu einem vermehrten Einsatz im „Embedded Systems“-Bereich führt. Die Kombination aus einem Prozessor mit Speicherkontroller und Peripherie sowie einer FPGA-Struktur in einem SoC zeigt Abbildung 1.1.

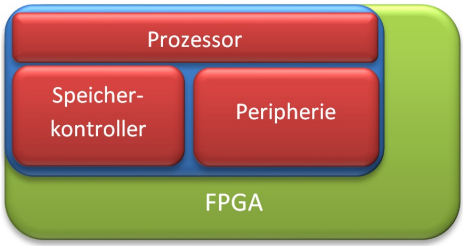

Abbildung 1.1: Prinzipbild eines SoCs

Verschlüsselungsalgorithmen laufen meist dann auf einem FPGA, wenn vor allem die Geschwindigkeit und der Durchsatz der Daten von Relevanz sind [MMSS02]. Anwendungen dazu findet man weitverbreitet in Konsumgütern wie Router, aber auch als Beschleuniger für Verschlüsselungen im Bereich der Industrie und des Militärs.

Moderne Fabriken entwickeln sich heute mehr und mehr in Richtung „Smart Factories“ - ein Begriff der „Industrie 4.0“, wie in [Plo14] beschrieben. Die „Industrie 4.0“ wird dabei als die vierte industrielle Revolution angesehen, da sie maßgebliche Änderungen in den Produktionsabläufen vorsieht. Mit Hilfe von verteilten Sensornetzwerken soll beispielsweise in einer intelligenten Produktionsanlage eine flexible, effiziente Wertschöpfungskette geschaffen werden. Diese verteilten, dezentralen Systeme - in Form von „Cyber-Physical Systems“ (CPS) - verwenden dabei oft unsichere Kommunikationskanäle, wie das Internet oder Funkkanäle, weshalb ein wirkungsvoller Schutz vor Manipulation und Sabotage verlangt wird. Das begründet die Anwendung von Verschlüsselungsimplementierungen für einen sicheren Datenaustausch in einem „Embedded System“.

Nicht nur dem Industriesektor steht ein Wandel bevor. Auch im privaten Bereich zeichnet sich ein Trend der umfassenden Vernetzung ab, der ebenfalls eine sichere verschlüsselte Übertragung der Daten benötigt. Alltagsgegenstände wie Kühlschrank, Auto oder Klimaanlage werden immer häufiger über das Internet für Steuerung, Wartungsarbeiten oder Softwareupdates miteinander verknüpft. Die veröffentlichten Hochrechnungen in [Eva11] zeigen, dass es heute mehr mit dem Internet verbundene Geräte als Menschen auf der Erde gibt. Diese Entwicklung der letzten Jahre hat den Begriff „Internet der Dinge“ (IoT - „Internet of Things“) geprägt. Die Grafik 1.2 zeigt den enormen Wachstum der letzten Jahre, sowie Prognosen für das Verhältnis zwischen mit dem Internet verbundenen Geräten und der Weltbevölkerung.

Diese neuen Perspektiven bringen aber auch viele Gefahren mit sich, denen man sich rechtzeitig stellen muss. In geographisch weit verbreiteten Netzen, wie es die Industrie 4.0 und IoT vorgeben,

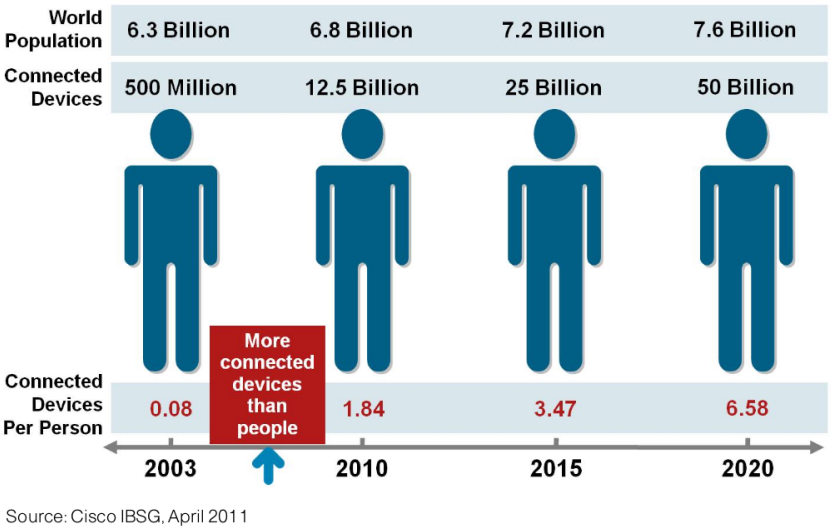

Abbildung 1.2: Verbundene Geräte mit dem Internet [Eva11]

kann ein physischer Zugriff auf einzelne Knoten oft nicht verhindert werden, wodurch es zu einer zusätzlichen Gefährdung in Bezug auf Seitenkanalangriffe kommen kann.

Neben dem Angriff auf Verschlüsselungsimplementierungen steht auch der Bruch der „Bitstream"-Verschlüsselungen [MBKP11] seit einigen Jahren im Fokus der Forscher. Eine solche Verschlüsselung soll verhindern, dass Unbefugte, während des Startvorgangs eines FPGA und dem damit verbundenen Laden eines „Bitstreams" aus einem externen Speicher, direkt auf die Konfigurationsdaten zugreifen können. Für die Authentifizierung der Konfiguration vor dem Laden wird dabei gerne auf asymmetrische Verschlüsselungen, wie es die Kryptographie über RSA möglich macht, gesetzt [4].

Um einen Angriff dieser Art zu verteidigen, existieren Gegenmaßnahmen, die eine Analyse von Seitenkanälen verhindern sollen. Jedoch verwenden viele der im Umlauf befindlichen Implementierungen keine dieser Maßnahmen gegen Seitenkanalangriffe, was mehrfach begründet werden kann:

- Seitenkanalangriffe werden oft unterschätzt oder sind dem Entwickler nicht bekannt.
- Viele Entwickler arbeiten unter Kosten- beziehungsweise Zeitdruck, was meist im Widerspruch zu einer geschützten Implementierung steht.
- Die Anforderung einer möglichst schnellen und optimierten Implementierung widerspricht einer Softwareumsetzung mit möglichst geringen Seitenkanalinformationen, da diese einen zusätzlichen rechnerischen Aufwand bedeuten.
- Das Testen von Seitenkanalgegenmaßnahmen ist aufwendig und der Erfolg eines Angriffs stark von der Messumgebung abhängig.

Der Schutz vor einem solchen Angriff wird im Gegensatz zu dem Angriff anhand von mathematischen Methoden, wie die Faktorisierung, nicht durch eine Erhöhung der Schlüssellänge verbessert. Es kann dabei sogar der gegenteilige Effekt in Kraft treten, der aus der zusätzlich verbrauchten

Logik in einem FPGA eine signifikantere Verlustleistungscharakteristik verursacht. Analog dazu, dass ein endlicher Schlüssel auf eine endliche mathematische Sicherheit hinweist, kann ein Seitenkanalangriff nicht durch Gegenmaßnahmen ausgeschlossen werden, sondern nur einen erfolgreichen Angriff deutlich erschweren.

1.2 Problemstellung

Die Umsetzung einer Verschlüsselung in einem FPGA kann im Allgemeinen auf verschiedenen Algorithmen basieren. Diese sind, abhängig von der Ressourcenverfügbarkeit und den Geschwindigkeitsanforderungen, mehr oder weniger leicht zu implementieren. Das bedeutet, dass aus verschiedenen Rechenvorgängen und Zwischenergebnissen dasselbe Verschlüsselungs- beziehungsweise Entschlüsselungsergebnis folgt. Aus diesem Grund muss für einen erfolgreichen Seitenkanalangriff zuerst der verwendete Algorithmus bekannt sein. Mit dieser Erkenntnis kann eine darauf folgende Seitenkanalanalyse adaptiert und optimiert werden, da sich verschiedene Implementierungen desselben Algorithmus nicht immer ganz ähnlich verhalten. Die Vorgangsweise für einen Angriff auf ein unbekanntes System ist in Grafik 1.3 ersichtlich.

Hersteller von Geräten die Verschlüsselungen ausführen, geben oft keine Implementierungsdetails bekannt, wodurch der Angreifer vor einer Art Blackbox steht. Um dieser Problematik zu begegnen, befasst sich diese Arbeit mit Methoden, die Informationen aus einer Blackbox sammeln. In der aktuellen Literatur findet man hauptsächlich Seitenkanalangriffe auf bekannte Systeme, wobei die Detektion der Algorithmen selbst keine große Rolle spielt. Daher soll sich diese Diplomarbeit mit dem Forschungsgebiet der Verschlüsselungen auf FPGAs befassen.

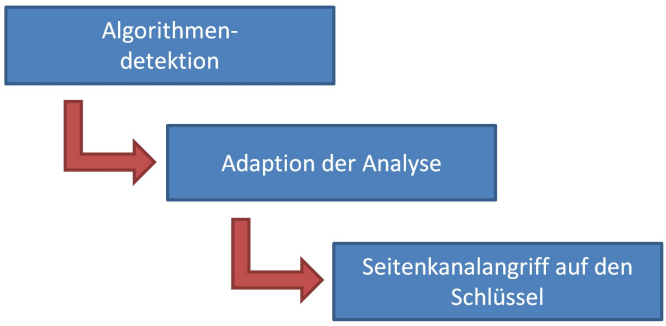

Abbildung 1.3: Vorgangsweise für einen Angriff auf ein unbekanntes System

1.3 Aufgabenstellung und Methodik

Ziel dieser Diplomarbeit ist es, eine Methodik zu schaffen, die ausgewählte Verschlüsselungsalgorithmen in einem unbekannten System anhand von verschiedenen Merkmalen aus einer FPGA-Blackbox Implementierungen detektieren kann. Die dazu nötigen Informationen sollen anhand von verschiedenen Seitenkanalangriffen gewonnen werden. Eine weitere Intention dieser Arbeit ist es, diese Implementierungen in einem praktischen Abschnitt mit den erarbeiteten Seitenkanalangriffen zu attackieren und den geheimen Schlüssel zu extrahieren. Um dies zu erreichen, muss eine Messumgebung geschaffen werden, die den Angriff erst möglich macht.

Neben dem Literaturvergleich zum Stand der Technik in Kapitel 2, behandelt diese Arbeit verschiedene Verschlüsselungsmethoden und legt einen besonderen Fokus auf RSA, welche sich im Hauptabschnitt 3 wiederfinden. Die genaue Analyse mathematischer Rechenvorgänge, wie dem chinesischen Restsatz, das binäre modulare Potenzieren oder die modulare Multiplikation im Ursprungs- beziehungsweise Montgomery-Raum, ist für die Interpretation der Seitenkanalmessung unumgänglich und gilt als notwendige Grundlage für einen Angriff. Das Konzept und genauere Informationen zu verschiedenen Seitenkanalangriffsvektoren - mehrere mögliche Angriffswege über Seitenkanäle - werden im anschließenden Hauptkapitel 4 behandelt.

Anschließend stellt Kapitel 5 eine Methodik vor, wodurch Seitenkanalangriffe in ihrer Kombination eine unbekannte Implementierung entlarven können. Für die Analyse werden drei verschiedene Seitenkanalangriffsvektoren verwendet: „Simple Power Analysis“ (SPA), „Differential Power Analysis“ (DPA) und die Rechenzeitanalyse. SPA und DPA basieren beide auf der, durch die Berechnung veränderte Verlustleistungsänderung im FPGA, wobei DPA mittels mehrerer Messaufnahmen die Beobachtungen statistisch verarbeitet. Anhand der Analyse der Rechenzeit eines Algorithmus ist es dem Angreifer möglich, interne Vorgänge durch die Zeitmessung der Gesamtausführung oder die zeitliche Beobachtung von Änderungen signifikanter Stellen in der Verlustleistung zu detektieren.

Das Kapitel 6 untersucht praktische Angriffe auf Verschlüsselungsalgorithmen in einem FPGA und zeigt wie aus beobachtbaren internen Vorgängen Schlüsselinformationen extrahiert werden können.

Abschließend beschreibt Abschnitt 7 die Modifikation einer FPGA Entwicklungsplatine, welche die Seitenkanäle für die Messung erst zugänglich machen. Zusätzlich wird über die Verifikation der Seitenkanäle anhand erster Untersuchungen und Tests berichtet.

Um sich vor den, in dieser Arbeit vorgestellten Angriffen zu schützen, werden in den Unterkapiteln 4.6 und 5.8 Gegenmaßnahmen und Lösungen diskutiert, die einerseits den beobachteten Seitenkanal und andererseits die Detektion des Algorithmus erschweren oder verhindern können.

2 Verwandte Arbeiten

Der Inhalt in diesem Kapitel widmet sich dem Stand der Technik in der Literatur, indem es verwandte Arbeiten analysiert und vergleicht. Da das Thema Seitenkanalangriff für sichere Implementierungen relevant ist, wurde es schon umfangreich untersucht, wobei die Detektion des Algorithmus nur wenig Beachtung findet. Neben den Vergleich zwischen dieser Diplomarbeit zu anderen Publikationen, wird außerdem auch auf potentielle Vor- und Nachteile eingegangen.

2.1 Erste Seitenkanalangriffe

Nach [Zef07, S. 6-7] galt Anfang der neunziger Jahre die Ausführung einer Verschlüsselungsmethode auf einem „Trusted System Modul“ - ein Chip der ein beliebiges Gerät mit Sicherheitsfunktionen ausstattet - als sicher. Die Veröffentlichungen der ersten Seitenkanalangriffe auf diese Systeme änderten diese Ansicht schlagartig.

Als Pionier dafür gilt Paul Carl Kocher, der 1996 seine Forschungsarbeit „Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems“ [Koc96] publizierte, die die Rechenzeit von verschiedenen Verschlüsselungsmethoden auf einer Smartcard analysiert. Er beschreibt die statistische Messung der Ausführungszeit und wie mithilfe dieser der Schlüssel bitweise extrahiert werden kann. Zudem nutzt er den Umstand, dass die verwendete arithmetische Logik während der modularen Multiplikation für manche Operanden mehr Zeit benötigt als für andere. Diese verwendete Angriffsmethode ist für eine modulare Multiplikation in einer FPGA Implementierung nicht relevant, da die Berechnungszeit von verschiedenen Operanden für eine modulare Multiplikation konstant gehalten werden kann. Seine Arbeit beinhaltet außerdem einen Rechenzeitangriff auf den optimierten Verschlüsselungsalgorithmus, der auf dem chinesischen Restsatz Theorem (CRT-RSA) beruht. Dieser Ansatz, der die Vorberechnungen für diesen Algorithmus nutzt, wird in der vorliegenden Diplomarbeit ebenfalls als Detektionsmethode eingesetzt.

Die beiden Seitenkanalangriffsvektoren SPA und DPA wurden erstmals im Jahr 1999 in der Literatur erwähnt. Die Publikation „Differential Power Analysis“ [KJJ99] beschreibt unter anderem den Angriff auf asymmetrische Verschlüsselungsmethoden und die besondere Stärke von statistischen Methoden in der Analyse. Die Arbeit geht davon aus, dass dem Angreifer Detailinformationen der Implementierung vorliegen, was für eine Blackbox-Analyse nicht zutrifft. Außerdem gehen die Autoren von computerähnlichen Strukturen aus, weswegen sich dieses verwandte Werk deutlich von dieser Arbeit abgrenzt.

Die beiden vorgestellten Arbeiten gaben den Anstoß, in den darauffolgenden Jahren weitere Forschungen in diesem Gebiet zu betreiben. Einerseits wurden neue Seitenkanäle erforscht und Angriffsvektoren für die bereits existierenden Seitenkanäle verfeinert, auf der anderen Seite verschiedene Verschlüsselungsmethoden attackiert.

2.2 Seitenkanalangriffe auf Field Programmable Gate Arrays

Erste experimentelle Seitenkanalanalysen auf Verschlüsselungsmethoden an einem FPGA wurden 2003 vorgestellt [ÖOP03]. Über SPA wird eine ECC Verschlüsselungsmethode attackiert. Eine Interpretation der Messergebnisse ist für diese Methode einfacher, als für eine RSA Verschlüsselung. Diese Eigenschaft ist auf die Unterscheidung der beiden ECC-Operationen Punkt-Addition und Punkt-Verdoppelung zurückzuführen, die durch ihren verschiedenen Aufbau leichter fällt als die beiden Operationen modulares Multiplizieren und Quadrieren in RSA. Der Messaufbau unterscheidet sich dahingehend, dass das FPGA auf einer Platine ohne zusätzlicher Peripherie arbeitet und sich somit eine deutlich vereinfachte Messumgebung ergibt. Das bietet außerdem die Möglichkeit, einzelne Versorgungspins der gleichen Versorgungsspannung am Board abzugreifen und zu vergleichen. Eine solch ideale Umgebung ist weit entfernt von einer realen Umgebung einer Blackbox, weswegen dieses Ziel nicht verfolgt wird. Genauere Hintergrundinformationen zu ECC finden sich im Anhangsabschnitt A.1.

Die beiden Arbeiten „Power analysis of FPGAs: How practical is the attack?“ [SOSQ03] und „Differential Power Analysis Attack on FPGA Implementation of AES“ [VY08] attackieren Verschlüsselungsalgorithmen auf einer FPGA Entwicklungsplatine mit Hilfe von DPA und einem Verlustleistungsmodell, das nur die Hamming Distanz betrachtet. Dieses Modell eignet sich für die Simulation einer Mikroprozessor-Architektur, jedoch nicht für die untersuchten Implementierungen auf einem FPGA. Neben der Hamming Distanz wird deshalb in dieser Diplomarbeit auch die Auswirkung von kurzzeitigen, logischen Fehlstellungen - verursacht durch unterschiedliche Laufzeiten von asynchronen Schaltteilen - auf die Verlustleistung untersucht und für das Modell verwendet.

Das „Japanese Research Center for Information Security“ (RCIS) hat in der Zusammenarbeit mit der Tohoku University in Japan gemeinsam verschiedene FPGA Entwicklungsboards für die Untersuchung von Seitenkanälen entwickelt. Auch das Nachfolgeprojekt SAKURA („Sidechannel Attack User Reference Architecture“) ist in einigen wissenschaftlichen Arbeiten als Untersuchungsplattform im Einsatz. Eine der aktuellsten Versionen davon ist unter der Bezeichnung „SAKURA-X“ beziehungsweise „SASEBO-GIII“ bekannt [5]. Abbildung 2.1 zeigt eine Platine, die mit einem Kintex-7 und einem Spartan-6 FPGA von XILINX ausgestattet ist.

Der große Vorteil, der die Verwendung dieser Boards mit sich bringt, liegt in den für die Seitenkanalanalyse herausgeführten Messanschlüssen und in der optimierten Messumgebung. Der Beobachter muss sich daher nicht um die Aufbereitung der Messsignale im Messaufbau kümmern. Die Entscheidung gegen den Einsatz einer solchen Seitenkanalmessplatine wurde aufgrund der starken Abweichungen im Vergleich zu einem praxisnahen Angriff getroffen. Außerdem sind die Anschaffungskosten für ein solches System deutlich höher als jene für ein herkömmliches Entwicklungsboard.

Viele wissenschaftliche Artikel, die Seitenkanäle auf FPGAs analysieren, verwenden FPGA Entwicklungsplattformen, die aus den beiden Projekten SAKURA oder SASEBO entstanden sind. Eine Arbeit davon nennt sich „Collision-Based Power Analysis of Modular Exponentiation Using

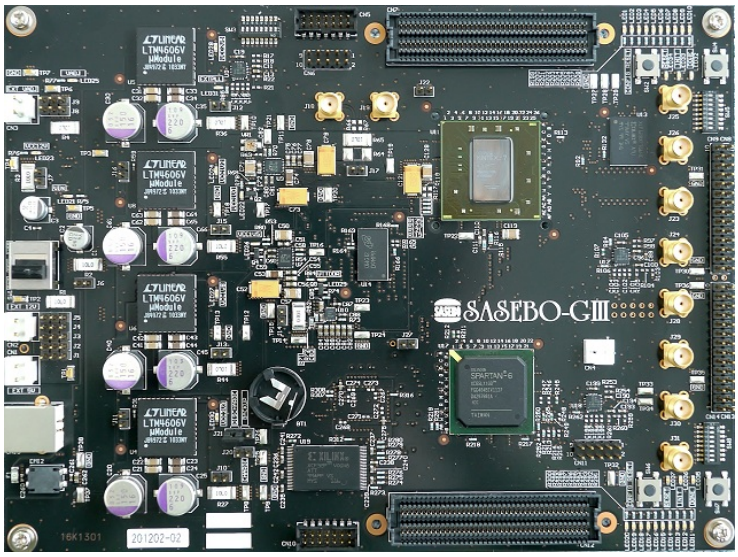

Abbildung 2.1: SASEBO-GIII Seitenkanalanalyse Testumgebung [5]

Chosen-Message Pairs“ [SSH+08] und beschreibt den Angriffsvektor „Comparative Power Analysis“, der versucht Schlüsselinformationen über Kollisionen zwischen zwei unterschiedlich vorgegebenen Nachrichten zu erlangen. Anhand der Verlustleistungsdifferenz zum selben Zeitpunkt der Entschlüsselung kann so zwischen den beiden chiffrierten Werten $-X$ und X ein hintereinander folgendes Quadrieren im binären modularen Potenzieren erkannt und in weiterer Folge der private Schlüssel daraus rekonstruiert werden. Diese Angriffsmethode weist starke Verwandtschaft zu der „Doubling Methode“ auf, die auch Kollisionen in der Berechnung verwendet. Kollisionsmethoden gelten als starkes Werkzeug, um Daten aus einer RSA Verschlüsselung zu extrahieren. Eine Kollision entsteht zum Beispiel dadurch, dass zwei Verschlüsselungsvorgänge mit einem beliebigen Wert X und dem dazugehörigen Wert -X geladen werden. In einem „Quadrieren und Multiplizieren“-Algorithmus folgt somit für eine binäre Null im Schlüssel zum selben Zeitpunkt dasselbe Verlustleistungsmuster. Dieser Ansatz ergibt jedoch den Nachteil, dass er unabhängig von der verwendeten Multiplikationseinheit funktioniert und deshalb für eine Blackbox-Analyse nicht sinnvoll ist.

Die Analyse auf einer „SASEBO-G“ Plattform beschreibt die Publikation „Enhanced Power Analysis Attack Using Chosen Message Against RSA Hardware Implementations“ [MHAS08]. Sie verwendet in ihren experimentellen Untersuchungen für den Angriffsvektor SPA Extremwerte. Dazu wird der Modulus, der einen Teil des öffentlichen Schlüssels repräsentiert, um eine Zahl reduziert. Die Forschungsfrage darin beschränkt sich jedoch auf eine Implementierung und die Schlüsselfindung und nicht auf die Demaskierung von verschiedenen Verschlüsselungsalgorithmen.

Die Stärken und Beschränkungen von elektromagnetischen Seitenkanalangriffen auf einem FPGA beschreibt die Arbeit „Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis“ [HMH+13]. Eine besondere Stärke in diesem Kanal ergibt sich vor allem durch die mögliche Granularität des Verfahrens. Damit diese Teilinformationen eingefangen werden können, muss der lokale Messaufbau sehr präzise gestaltet sein. Außerdem ist dafür eine große Anzahl an Messungen nötig, die das starke Rauschen reduzieren. Vergleicht man den Seitenkanal der elektrischen Verlustleistung mit dem der elektromagnetischen Abstrahlung, so zeigt sich, dass der Erfolg von elektromagnetischen Attacken von der Vorverarbeitung der Messdaten abhängig ist. Die elektrische Verlustleistung kann hingegen einfacher aufgenommen

werden und repräsentiert zu jedem Zeitpunkt die aktuelle Beobachtung in einem Wert.

Die Publikation „Dynamic Partial based Single Event Upset (SEU) Injection Platform on FPGA“ [GM13] stellt eine Möglichkeit vor, Fehler im Konfigurationsspeicher eines FPGA zu injizieren. Weiters zählt sie die Nachteile auf, die in einer solchen typischen fehlerinjizierenden Methode auftreten können. Dieser Angriffsvektor kommt für eine Blackbox-Analyse deshalb nicht in Frage, da diese Methode in ihrer Auswirkung nicht genau kontrolliert werden kann und in ihrer Ausführung nicht immer ganz zerstörungsfrei arbeitet.

2.3 Detektion von Algorithmen

Alle in den beiden vorherigen Abschnitten erwähnten Werke zielen auf die Analyse von bekannten Systemen ab und nicht auf die Detektion von unbekannten Algorithmen. Unbekannte Verschlüsselungssysteme werden in der heutigen Literatur nur wenig betrachtet. Hinter dem Akronym SCARE verbirgt sich der Begriff „Sidechannel Analysis Reverse Engineering“, der für die Blackbox-Analyse steht. Die Autoren beschreiben dazu in ihrer Publikation „Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms“ [AFV07] eine Möglichkeit, um Verschlüsselungsvorgänge auf Co-Prozessoren in Smartcards zu erkennen und einem Verschlüsselungsalgorithmus zuzuordnen. Sie bedienen sich dabei der „Correlation Power Analysis“ und einem Hardware Simulationsmodell, das auf der Hamming Distanz basiert. Durch die vielen Freiheiten der parallelen Implementierungsvarianten auf einem FPGA setzt diese Vorgangsweise für die Detektion eines unbekannten Designs Detailinformationen oder eine große Modelldatenbank voraus.

Ein ähnliches Problem tritt für Template Analysen auf unbekannte Algorithmen auf. Im Gegensatz zu Mikrocontrollern, so wie es in der Arbeit „Building a Side Channel Based Disassembler“ [EPW11] beschrieben ist, sind solche Methoden für einen FPGA eher ungeeignet. Eine Implementierung in einem hardwareprogrammierbaren Gerät kann auf viele Arten umgesetzt und nur bei genauer Kenntnis in einem Template abgebildet werden. Deswegen beschreiben die in dieser Diplomarbeit vorgestellten Methoden einen generischen SCARE-Ansatz, der auch Gültigkeit auf FPGAs hat.

2.4 Gegenmaßnahmen

Blackbox-Analysemethoden verwenden für die Detektion oft verwandte Prinzipien der herkömmlichen Seitenkanalangriffe. Dies führt dazu, dass oftmals dieselben Gegenmaßnahmen einsetzbar sind. Paul Kocher, Benjamin Jun, Joshua Jaffe und Pankaj Rohatgi beschreiben in ihrer Arbeit „Introduction to Differential Power Analysis“ [KJJR11] allgemeine Schutzmechanismen, die die Auswirkungen des Reduzierens von Signal-Rausch-Verhältnissen untersuchen. Außerdem wird das Verstecken und Maskieren von Verschlüsselungsvorgängen aufgezeigt. Die Effektivität von Maßnahmen gegen die Analyse des implementierten Verschlüsselungsalgorithmus unterscheidet sich von einem Seitenkanalangriff auf den geheimen Schlüssel und wird deshalb in dieser Arbeit diskutiert.

Die Auswirkung von umgesetzten Gegenmaßnahmen in einem FPGA zeigt der Artikel „RSA Power Analysis Obfuscation: A Dynamic FPGA Architecture“ [Bar12]. Dieser setzt auf dynamische Algorithmen, die die Ausführung randomisieren. Die Bedeutung des gemessenen Seitenkanals

muss bei einem Angriff auf diesen Schutzmechanismus neu interpretiert werden. Im Gegensatz dazu erhöht sich die Komplexität der Messung einer Verschlüsselungsimplementierung, die durch spezielle Maßnahmen das Signal-Rausch-Verhältnis (SNR) im Seitenkanal verschlechtert, kaum. Das liegt daran, dass durch erhöhten statistischen Messaufwand das SNR wieder verbessert werden kann.

3 Verschlüsselungsmethoden

Durch eine Vielzahl von Verschlüsselungsverfahren kann Klartext durch Geheimtext ersetzt werden. Asymmetrische Verschlüsselungen, auch unter dem Namen Public-Key-Verfahren bekannt, unterscheiden sich zu symmetrischen Verschlüsselungsmethoden dahingehend, dass pro kommunizierender Partei jeweils ein öffentlicher und ein privater Schlüssel existieren und dadurch ein einfacher ungesicherter öffentlicher Schlüsselaustausch stattfinden kann. In der breiten Anwendung finden sich heutzutage vor allem die beiden asymmetrischen Systeme RSA und „Elliptic Curve Cryptography“ (ECC) wieder.
Dieses Kapitel versucht einen Einblick in die Arbeitsweisen des Verschlüsselungsverfahren RSA und verschiedener Implementierungen zu schaffen, sodass darauf basierende Seitenkanalangriffe verständlich gemacht werden können. Letztlich steht im Unterkapitel 3.3 die Anwendung von RSA im Mittelpunkt der Betrachtung.

3.1 Mathematische Grundlagen

Bevor im folgenden Unterkapitel auf verschiedene kryptographische Algorithmen und Methoden eingegangen wird, versucht dieser Abschnitt notwendige mathematische Grundlagen zu schaffen.

Der kleine Satz von Fermat

Der aus der Zahlentheorie bekannte Lehrsatz von Fermat hat hohe Relevanz für die Funktionsweise der untersuchten Verschlüsselung. Für eine beliebige Primzahl p und eine Zahl k zwischen 1 und p gilt [Kob94, S. 20]:

$$k = k^p \mod p \tag{3.1}$$

beziehungsweise

$$1 = k^{p-1} \mod p \tag{3.2}$$

Eulersche φ - Funktion

Diese Funktion wird in [Len07, S. 224] beschrieben und bestimmt die Anzahl der positiven ganzen Zahlen, die kleiner und gleichzeitig teilerfremd zum übergebenen Parameter n sind. Bei dem Parameter muss es sich ebenfalls um eine positive ganze Zahl handeln. Zwei Zahlen sind zueinander

teilerfremd, wenn der größte gemeinsame Teiler, kurz ggT, eins ergibt. Mit dem Mengenoperator # lässt sich diese Funktion wie folgt zusammenfassen:

$$\varphi(n) := \#\{m \in \mathbb{N}^* | ((0 < m < n) \wedge (ggT(m, n) = 1))\} \tag{3.3}$$

Die eulersche Funktion mit einer Primzahl p als Argument berechnet sich demnach immer zu $\varphi(p) = (p - 1)$, da eine Primzahl zu einer kleineren Zahl keinen gemeinsamen Teiler aufweist.

Satz von Euler

Dieser Satz, der sich als die Verallgemeinerung des kleinen Satzes von Fermat ergibt, sagt nach [Eck11, S. 353-354] aus, dass die Kongruenz

$$a^{\varphi(n)} \equiv 1 \mod N \tag{3.4}$$

gültig ist, wenn a und n eins als den größten gemeinsamen Teiler besitzen. Der Exponent stellt sich dabei als die eulersche Funktion $\varphi(n)$ heraus.

Diskretes logarithmisches Problem

Laut [HPS08, S. 62-65] versteht man unter der Abkürzung DLP in der Kryptographie das diskrete logarithmische Problem, welches die Grundlage für viele Verschlüsselungsmethoden darstellt (ECC, Elgamal, Diffie-Hellman-Schlüsselaustausch). Es beschreibt die Problematik einen Exponenten x zu finden, sodass

$$g^x \equiv h \mod p \tag{3.5}$$

gilt, wobei g eine ganze Zahl zwischen null und p darstellt. Gibt es für $x = \log_g(h)$ eine Lösung, so existieren, aufgrund des kleinen Satzes von Fermat, unendlich viele verschiedene Ergebnisse für x, die die Gleichung 3.5 erfüllen.

Faktorisierungsproblem

Durch das Faktorisieren einer Zahl wird diese in seine multiplikativen Faktoren zerlegt. Nach [MO96, S. 89-90] wird diese Aufgabe nicht-trivial, wenn die zu zerlegende Zahl aus dem Produkt von zwei unbekannten, großen, zueinander teilerfremden Zahlen besteht. Der öffentliche Teilschlüssel N eines RSA-Verschlüsselungsvorgangs besteht aus einem solchen geheimen Produkt. Aufgrund dessen hängt die Sicherheit der genannten Chiffrierung stark von der Lösung dieses Problems ab.

3.2 RSA – Algorithmus

Die Entwicklung der RSA-Verschlüsselung fand im Jahr 1977 statt, wobei sich der Name aus den Anfangsbuchstaben seiner Erfinder (Rivest, Shamir und Adleman) ableitet [KK10, S. 111]. Es handelt sich dabei um eine Blockverschlüsselung in Form eines Public-Key-Verschlüsselungsverfahrens. Wie in [Pog07, S. 86-90] erklärt, basiert RSA auf der Grundlage einer Falltürfunktion $f(x)$. Das bedeutet, dass der Funktionswert $y = f(x)$ mit der Kenntnis des Parameters x einfach berechnet

werden kann. Es ist jedoch schwierig anhand des Wissens des Funktionswerts y auf den Parameter x rückzuschließen. Diese Umkehrung funktioniert nur mit einer geheimen Information, dem privaten Schlüssel. Das modulare Potenzieren für große natürliche Zahlen, wie es in RSA vorkommt, stellt so eine Falltürfunktion dar.

Die folgenden Schritte zeigen die Schlüsselgenerierung mit anschließender Ver- und Entschlüsselung [Eck11, S. 353]:

1. Zuerst müssen zwei große Primzahlen p und q gewählt und der RSA-Modulus $N = p \cdot q$ berechnet werden. Dieser berechnete Wert gibt den möglichen Schlüsselraum und somit die verwendete Schlüsselbreite an. Die beiden Primzahlen bestimmt man üblicherweise so, dass sie die gleiche Bitbreite aufweisen. Das hat den Hintergrund, dass bei einem Angriff über die Faktorisierung von N die kleinere der beiden leichter gefunden werden kann.

2. Die eulersche Funktion zeichnet sich für den Modulus N, welcher aus zwei teilerfremden Zahlen besteht, als $\varphi(N) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$ ab.

3. Im nächsten Schritt legt der Schlüsselerzeuger einen zufälligen Wert d zwischen eins und $\varphi(N)$ fest, sodass gilt: $ggT(\varphi(N), d) = 1$.

4. Infolgedessen wird der Wert e bestimmt, der sich aus der Kongruenz $e \cdot d \equiv 1 \mod \varphi(N)$ berechnen lässt und sich als die multiplikative Inverse von Modulo $\varphi(N)$ zu d herausstellt.

5. Die Verschlüsselung eines Klartexts $M \in [0, N-1]$ erfolgt über die Gleichung

$$C = M^e \mod N. \tag{3.6}$$

6. Die Umkehrung eines verschlüsselten Texts $C \in [0, N-1]$ berechnet sich mit

$$M = C^d \mod N. \tag{3.7}$$

Aus diesen beiden letzten Schritten wird deutlich, dass es sich bei dem Schlüsselpaar (e, N) um den öffentlichen Schlüssel und bei dem Wert d um den privaten Schlüssel handelt.

Um die Funktionsweise von RSA besser zu verstehen, soll die Herleitung der Verschlüsselungsmethode aus [Eck11, S. 354-355] helfen. Zu zeigen ist, dass folgende Formel gilt:

$$M \equiv M^{e \cdot d} \mod N \tag{3.8}$$

Es werden dabei drei Fälle unterschieden:

Fall 1: Im ersten Fall sind die zwei Primzahlen p und q kein Teiler von M. (Diese Bedingung ist gleichbedeutend mit $ggT(p, M) = 1$ beziehungsweise $ggT(q, M) = 1$.)
Die Voraussetzung $e \cdot d \equiv 1 \mod \varphi(n)$ kann auch als Gleichung

$$e \cdot d = 1 + k \cdot \varphi(n) \tag{3.9}$$

mit $k \in \mathbb{N}$ angeschrieben werden. Deswegen gilt:

$$M^{e \cdot d} = M^{1 + k \cdot \varphi(n)} = M \cdot M^{k \cdot (p-1) \cdot (q-1)} \tag{3.10}$$

Mit Hilfe des kleinen Satzes von Fermat (Kapitel 3.1) ist es möglich diese beiden Kongruenzen zu definieren:

$$M \cdot \underbrace{M^{(p-1)^{k\cdot(q-1)}}}_{=1} \equiv M \mod p \tag{3.11}$$

$$M \cdot \underbrace{M^{(q-1)^{k\cdot(p-1)}}}_{=1} \equiv M \mod q \tag{3.12}$$

Weil $M^{e\cdot d}$ durch p und q teilbar ist und $ggT(p,q) = 1$ gilt, teilt auch das Produkt $N = p \cdot q$ diese Zahl.

$$M \equiv M^{e\cdot d} \mod N \tag{3.13}$$

Fall 2: Wenn M ein Vielfaches von p beinhaltet, dann tritt dieser Beweisabschnitt ein. Aufgrund dieser Vorbedingungen kann M folgendermaßen angeschrieben werden:

$$M = i \cdot p,\ i \in \mathbb{N} \tag{3.14}$$

$$M \mod p = 0 \tag{3.15}$$

$$\underbrace{(i \cdot p)^{e\cdot d}}_{=0} \equiv M \mod q \tag{3.16}$$

Durch die RSA-Voraussetzung $M \in [0, N-1]$ ist es nicht möglich, dass beide Primzahlen p und q gleichzeitig M teilen können. (Mit der Ausnahme von Fall 3) Deswegen wird q und der letzte Rechenschritt wie im Fall 1 berechnet. Der analoge Beweis gilt auch für q.

Fall 3: Der letzte triviale Fall tritt ein, wenn die Nachricht und dadurch auch die verschlüsselte Zahl Null ist.

$$\underbrace{(M)^{e\cdot d}}_{=0} \equiv M \mod N \tag{3.17}$$

Die Stärke dieser Verschlüsselung liegt in der ineffektiven Faktorisierung von großen Zahlen (siehe Faktorisierungsproblem in Kapitel 3.1). Heute werden zufällig generierte Schlüssel mit einer Länge von mindestens 2000 Bits für die nächsten Jahre als sicher angesehen [BSI14, S. 36-37]. Den aktuellen Weltrekord im Primfaktorzerlegen erreichte Ende 2009 ein internationales Team in der Zusammenarbeit mit der Universität in Bonn [7], der einer RSA Bitbreite von 768 Bit entspricht.

Die Ver- und Entschlüsselung kann in verschiedenen Algorithmen erfolgen. Dabei sollte bedacht werden, dass es nicht möglich ist, zuerst die ganze Zahl zu potenzieren und anschließend den Rest zu berechnen. Die Begründung liegt darin, dass bereits bei kleineren Schlüssellängen, zum Beispiel eine 100 Bit breite Zahl, das Ergebnis die Anzahl der Atome im Universum (10^{89}) überschreitet. Ein effizienter Algorithmus hat aus diesem Grund die Aufgabe, ständig die Zwischenergebnisse zu beschränken.

3.2.1 Chinesischer Restsatz

Um eine gewisse Sicherheit zu erlangen, sollte der private Schlüssel eine gewisse Größe besitzen. Dieser Umstand führt dazu, dass daraus resultierende Rechenvorgänge mit sehr langen Zahlen viel Zeit in Anspruch nehmen. Das Theorem des chinesischen Restsatzes (kurz CRT) wird im Buch [KSRHP07, S. 133-134] präsentiert und erlaubt das modulare Potenzieren in zwei kleinere

Teile zu zerlegen, die auf den Primfaktoren p und q basieren. Diese Methode kann dabei nur zum Entschlüsseln und zum Signieren von Nachrichten verwendet werden, nicht aber für die Verschlüsselung der Daten mit dem öffentlichen Schlüssel. Wie bereits in Kapitel 3.2 erläutert, wird mit dieser Formel der chiffrierte Text in Klartext umgewandelt:

$$M = C^d \mod (p \cdot q) \tag{3.18}$$

Infolge des kleinen fermatischen Satzes, definiert im Abschnitt 3.1, kann

$$M_1 = C^d \mod p = C^{d_1 + z \cdot (p-1)} \mod p \quad \text{mit} \quad d_1 = d \mod (p-1),\ z \in \mathbb{N} \tag{3.19}$$

und

$$M_2 = C^d \mod q = C^{d_2 + w \cdot (q-1)} \mod q \quad \text{mit} \quad d_2 = d \mod (q-1),\ w \in \mathbb{N} \tag{3.20}$$

durch

$$M_1 = C^{d_1} \mod p \tag{3.21}$$

und

$$M_2 = C^{d_2} \mod q \tag{3.22}$$

vereinfacht werden.

Mit dem sogenannten „Mixed-Radix Conversion“ Ansatz [KSRHP07, S. 134-136] lässt sich in weiterer Folge das Ergebnis, welches sich als die entschlüsselte Nachricht herausstellt, kalkulieren:

$$M = M_1 + [(M_2 - M_1) \cdot (p^{-1} \mod q) \mod p] \cdot p \tag{3.23}$$

Die Gültigkeit dieses Vorgangs kann einfach über die Berechnung der beiden Modulo-Operationen auf die letzte Gleichung 3.23

$$M \mod p = M_1 \tag{3.24}$$

$$M \mod q = M_2 \tag{3.25}$$

und den Abgleich mit den beiden Formeln 3.19 und 3.20 gezeigt werden.

Der große Vorteil dieses Algorithmus liegt in der effektiven Implementierungsmöglichkeit, da er einerseits Rechenzeit für die Entschlüsselung spart und zum anderen im Vergleich zu einer modularen Exponentialfunktion mit voller Bitbreite einen geringeren Logikaufwand in einer rekonfigurierbaren Hardware verursacht.

3.2.2 Modulares Potenzieren

Die Berechnung von großen natürlichen modularen Potenzen, wie es besonders die Entschlüsselung eines chiffrierten Texts mit RSA vorsieht, kann durch verschiedene mathematische Vorgangsweisen auf einem FPGA realisiert werden. Die in diesem Unterkapitel vorgestellten Methoden geben Einblicke in einige ausgewählte Algorithmen.

3.2.2.1 Binäres modulares Potenzieren

Einer dieser Algorithmen nennt sich „binäres modulares Potenzieren“ und ist auch unter dem Namen „Quadrieren und Multiplizieren“ bekannt [MO96, S. 615]. Der Algorithmus 3.1 beschreibt diese mögliche Implementierung in der der Schlüssel von links nach rechts binär abgearbeitet wird.

Algorithmus 3.1 : Binäres modulares Potenzieren von links nach rechts

Eingabe:
$e = [e_{w-1}, e_{w-2}, \ldots, e_1, e_0] \ldots$ Binäre Darstellung des Exponenten (w Bit)
$x \ldots$ Basis
$N \ldots$ Modulus
Ausgabe: $y = x^e \mod N$

1: $y = x$
2: **for** $i = w - 1 \rightarrow 0$
3: $\quad y = y \cdot y \mod N$ $\quad \rightarrow$ Quadrieren
4: $\quad$ **if** $e[i] \neq 0$ **then**
5: $\quad\quad y = y \cdot x \mod N$ $\quad \rightarrow$ Multiplizieren
6: $\quad$ **end if**
7: **end for**

Um dieses Codestück besser zu verstehen, soll das folgende Beispiel, welches der Einfachheit halber auf die Modulo-Operation verzichtet, helfen. Dabei wird die Potenz x^{26} beziehungsweise in binärer Darstellung x^{11010b} für eine beliebige Zahl x berechnet. Durch die bitweise Betrachtung des Exponenten steht eine binäre Eins im mathematischen Ablauf für Quadrieren und Multiplizieren und eine Null für Quadrieren alleine.

1: $y = 1$ $\rightarrow$ Initialisierung
2: $y = x = x^1(x^{1b})$ $\rightarrow$ Multiplizieren
3: $y = y \cdot y = x^2(x^{10b})$ $\rightarrow$ Quadrieren
4: $y = y \cdot x = x^3(x^{11b})$ $\rightarrow$ Multiplizieren
5: $y = y \cdot y = x^6(x^{110b})$ $\rightarrow$ Quadrieren
6: $y = y \cdot y = x^{12}(x^{1100b})$ $\rightarrow$ Quadrieren
7: $y = y \cdot x = x^{13}(x^{1101b})$ $\rightarrow$ Multiplizieren
8: $y = y \cdot y = x^{26}(x^{11010b})$ $\rightarrow$ Quadrieren

Die Berechnung kann aber auch die einzelnen Bits des privaten Schlüssels von rechts nach links betrachten [MO96, S. 614]. Diesen Weg zeigt der Algorithmus 3.2.

Algorithmus 3.2 : Binäres modulares Potenzieren von rechts nach links

Eingabe:
$e = [e_{w-1}, e_{w-2}, \ldots, e_1, e_0] \ldots$ Binäre Darstellung des Exponenten (w Bit)
$x \ldots$ Basis
$N \ldots$ Modulus
Ausgabe: $y = x^e \mod N$

1: $y = x$
2: **for** $i = w - 1 \rightarrow 0$
3: $\quad$ **if** $e[i] \neq 0$ **then**
4: $\quad\quad y = y \cdot x \mod N$ $\quad \rightarrow$ Multiplizieren
5: $\quad$ **end if**
6: $\quad$ **if** $i \neq 0$ **then**
7: $\quad\quad y = y \cdot y \mod N$ $\quad \rightarrow$ Quadrieren
8: $\quad$ **end if**
9: **end for**

3.2.2.2 m-ary Methode

[Wan00, S. 65-67] verweist auf die m-ary Berechnungsvorschrift, die eine generalisierte Methode des „Quadrieren und Multiplizieren“ darstellt und vom Algorithmus 3.3 beschrieben wird. Danach funktioniert die Unterteilung des Exponenten e nicht nur bitweise, sondern kann auch in d w-Bit breiten Fenster erfolgen.

Algorithmus 3.3 : m-ary Methode

Eingabe:

$r = \log_2 m \ldots$ Bit-Fensterbreite
$m = 2^r$
$e = [e_{w-1}, e_{w-2}, \ldots, e_1, e_0] \ldots$ Binäre Darstellung des Exponenten (w Bit)
$s = w/r \ldots$ Anzahl der Blöcke
$F[i] \ldots$ i-ter Block mit m Bit des Exponenten
$x \ldots$ Basis
$N \ldots$ Modulus

Ausgabe: $y = x^e \mod N$

1: **for** $k = 2 \rightarrow m - 1$
2: Speichere $x^k \mod N$ → Vorberechnung für Schritt 2
3: **end for**
4: $y = x^{F[s-1]} \mod N$ → Initialisieren
5: **for** $i = s - 2 \rightarrow 0$
6: $y = y^m \mod N$ → Schritt 1
7: **if** $F[i] \neq 0$ **then**
8: $y = y \cdot x^{F[i]} \mod N$ → Schritt 2
9: **end if**
10: **end for**

Auch liefert [Wan00] ein Beispiel für diese generalisierte Methode, die sogenannte „Quaternary“ Methode. In dieser werden die Bits des Exponenten zu je zwei Bitwörter geteilt. Ein konkretes Beispiel mit Zahlen soll wieder die Funktionsweise veranschaulichen.

$$e = 506_{10} = \underbrace{01_2}_{F[4]} \underbrace{11_2}_{F[3]} \underbrace{11_2}_{F[2]} \underbrace{10_2}_{F[1]} \underbrace{10_2}_{F[0]} \tag{3.26}$$

$$s = \frac{w}{r} = \frac{10}{2} \tag{3.27}$$

Aufgrund dessen, dass die Modulo-Operation den Algorithmus verkompliziert, wurde sie nicht mitkalkuliert. Die einzelnen Schritte der Berechnung $y = x^{506} \mod N$ präsentieren die neun kommentierten Rechenvorgänge:

1: $y = x^{F[4]} = x^1$ → Initialisierung
2: $y = y^4 = x^4$ → Schritt 1
3: $y = y \cdot x^{F[3]} = x^7$ → Schritt 2
4: $y = y^4 = x^{28}$ → Schritt 1
5: $y = y \cdot x^{F[2]} = x^{31}$ → Schritt 2
6: $y = y^4 = x^{124}$ → Schritt 1
7: $y = y \cdot x^{F[1]} = x^{126}$ → Schritt 2
8: $y = y^4 = x^{504}$ → Schritt 1
9: $y = y \cdot x^{F[0]} = x^{506}$ → Schritt 2

Dem Vorteil dieser Berechnungsmethode, dass mehrere Bits gleichzeitig betrachtet und somit schneller abgearbeitet werden können, steht die Vorberechnung für den Schritt 2 gegenüber, die zusätzliche Logik in einem FPGA fordert.

Die m-ary Methode basiert auf dem Grundsatz, dass Bit-Fenster mit konstanter Länge an den im Vorhinein definierten Stellen unabhängig vom Exponenten existieren. Im Gegensatz dazu sucht die „Sliding Window"-Methode [Wan00, S. 68-71] nach Bit-Fenstern an Stellen, die nicht mit ausschließlich Nullen gefüllt sind und kann variable Fenstergrößen festlegen. Dies kann zu einer weiteren Beschleunigung der Berechnung führen.

Neben den bereits erwähnten Möglichkeiten gibt es noch anderer Varianten, wie zum Beispiel die „Factor"- und die „Power Tree"-Methode, die wegen ihres Aufbaus nur für kleinere Schlüssellängen praktikabel sind [KK94, S. 20-21].

3.2.3 Modulare Multiplikation

Wie im vorherigen Abschnitt 3.2.2 erkennbar ist, benötigt man für das binäre modulare Potenzieren Quadrier- und Multiplizier-Operationen. Deswegen beschäftigt sich dieses Kapitel genauer mit Algorithmen für die Berechnung der Form $y = a \cdot b \mod N$ wobei die Operatoren a, b und N Elemente der ganzen Zahlen sind.

Die primitivste Möglichkeit um eine modulare Multiplikation durchzuführen, besteht darin, zuerst die zwei k Bit langen Zahlen zu multiplizieren und das Ergebnis, welches aus einer $2 \cdot k$ Bit Zahl besteht, mit der Modulo-Operation wieder auf einen k Bit Wert zu reduzieren.

Bei dem Einsatz von langen Registervektoren ist zu beachten, dass sich nicht alle Operatoren aus einer VHDL-Bibliothek in einem FPGA umsetzen lassen. Zum Beispiel benötigt eine Implementierung, die auf die von der IEEE („Institute of Electrical and Electronics Engineers") herausgegebene VHDL-Bibliothek „NUMERIC_STD.ALL" zugreift, für Additionen, Subtraktionen, Vergleichs- oder Shift-Operatoren viel weniger Logik als für Multiplikations- und Modulo-Operatoren. Dieser erhöhte Logikverbrauch kommt daher, dass diese Operatoren in einem Takt ausgeführt werden und ihr Ressourcenaufwand stark mit der Erhöhung der Bitlänge ansteigt. Genau aus diesem Grund wird bei einer Verschlüsselungsimplementierung in einem FPGA versucht die modulare Multiplikation in mehrere Taktzyklen aufzuteilen und auf ressourcenaufwändige Operatoren zu verzichten.

3.2.3.1 Blakley-Multiplikation

Eine Rechenvorschrift um die modulare Multiplikation $a \cdot b \mod N$ zu berechnen, stellt die Blakley-Multiplikation in der Publikation [Bla83] dar. Diese „Bit-By-Bit" Multiplikation zeigt der Algorithmus 3.4 mit der Annahme, dass $0 \leq a, b \leq N - 1$ ist. Der Schritt „Bitweise multiplizieren" in diesem Pseudocode leitet sich aus der Umformung des Produktes in

$$a \cdot b = \sum_{i=0}^{k-1} (a_i \cdot b) \cdot 2^i \tag{3.28}$$

ab. Außerdem ist darin sichtbar, dass hier in jeder Iteration in der Schleife eine Schiebe-Operation und optional, abhängig vom Operand a, eine Addition durchgeführt werden muss.

Algorithmus 3.4 : Blakley-Multiplikation

Eingabe:

$a = [a_{k-1}, a_{k-2}, \ldots, a_1, a_0] \ldots$ k Bit breiter Operand

$b = [b_{k-1}, b_{k-2}, \ldots, b_1, b_0] \ldots$ k Bit breiter Operand

$N \ldots$ Modulus (wobei gilt $a, b < N$)

Ausgabe: $y = a \cdot b \mod N$

```
1:  y = 0
2:  for i = 0 → k − 1
3:      y = 2 · y + a_{k−1−i} · b        → Bitweise multiplizieren
4:      if y ≥ N then
5:          y = y − N                    → Restbildung Teil 1
6:      end if
7:      if y ≥ N then
8:          y = y − N                    → Restbildung Teil 2
9:      end if
10: end for
```

Dem schrittweisen binären Multiplizieren in Zeile 3 folgen zwei optionale Subtraktionsvorgänge, die das Ergebnis kleiner als das Modulus N halten. Die Subtraktion in der Restbildung muss zweimal durchgeführt werden, da das Zwischenergebnis y durch die bitweise Multiplikation einen Wert bis zu $3N - 3$ annehmen kann.

3.2.3.2 Multiplikation im Montgomery-Raum

Die Arbeit [KK94] beschreibt den mathematischen Hintergrund der Montgomery-Multiplikation, welche in diesem Abschnitt vorgestellt wird. Diese Rechenmethode zielt auf die Optimierung der modularen Multiplikation $a \cdot b \mod N$ für große ganze Zahlen ab, wobei die beiden Vorbedingungen $a < N$ und $b < N$ gelten müssen. Diese Verbesserung wird nur dann tragend, wenn mehrere modulare Multiplikationen hintereinander abzuarbeiten sind. Das hat den Hintergrund, dass aus der notwendigen Transformation in einen anderen Zahlenraum ein zusätzlicher Aufwand folgt, der eine Verbesserung für eine einzelne Multiplikation zunichtemacht. Ein sinnvoller Einsatz findet sich deswegen im bereits vorgestellten „Quadrieren und Multiplizieren"-Algorithmus im Kapitel 3.1. Hier wird die zusätzliche Transformation nur am Anfang und am Ende benötigt.

Wie bereits erwähnt, findet im ersten Schritt die Transformation in den sogenannten Montgomery-Raum statt. Dazu werden die beiden transformierten Multiplikatoren $\overline{a}$ und $\overline{b}$ durch eine modulare Multiplikation mit einer Zahl der Form $R = 2^m$, mit $m \in \mathbb{N}$, folgendermaßen berechnet [KK94, S. 46-49]:

$$\overline{a} = a \cdot R \mod N \tag{3.29}$$

$$\overline{b} = b \cdot R \mod N \tag{3.30}$$

Diese Transformationszahl R ergibt sich als nächstgrößere Zweierpotenz zu dem Modulus N. Dazu ist zu sagen, dass sich nicht jede modulare Multiplikation im Montgomery-Raum lösen lässt, da es keinen natürlichen gemeinsamen Teiler für die beiden Zahlen R und N geben darf, außer der Zahl Eins ($ggT^1(R, N) = 1$). Das heißt, die Bedingung gilt niemals für einen geraden

[1] größter gemeinsamer Teiler

Wert von N. Mit einem Trick über den chinesischen Restsatz, beschrieben im Abschnitt 3.2.1, gibt es die Möglichkeit ein gerades N in zwei ungerade Modulo-Berechnungen zu verwandeln. Das Ergebnis $\overline{c}$ des Produktes im Montgomery-Raum der beiden transformierten Werte $\overline{a}$ und $\overline{b}$, in dieser Arbeit mit dem Symbol $\circ$ abgekürzt, befindet sich ebenfalls im transformierten Raum und muss deswegen um den Transformationsfaktor R reduziert werden. Dazu verwendet man die modulare Inverse R^{-1} von R im Modulus N.

$$\overline{c} = \overline{a} \circ \overline{b} = \overbrace{(a \cdot R \mod N)}^{\overline{a}} \cdot \overbrace{(b \cdot R \mod N)}^{\overline{b}} \cdot R^{-1} \mod N = c \cdot R \mod N \tag{3.31}$$

Da die Multiplikation mit R^{-1} das Ergebnis in seiner Größe begrenzt, wird keine aufwendige Modulo-Operation in der Umsetzung benötigt. Der inverse Transformationsvektor besteht aus einer Zweierpotenz und kann dadurch einfach mit dem Einsatz einer Schiebe-Operation realisiert werden.

Ein Beispiel für das Potenzieren nach dem Schema des „Quadrieren und Multiplizieren“ spiegelt der Algorithmus 3.5 wider [KK94, S. 48].

Algorithmus 3.5 : Potenzieren im Montgomery-Raum

Eingabe:
$d = [d_{k-1}, d_{k-2}, \ldots, d_1, d_0]$ ... Binäre Darstellung des Exponenten (k Bit)
c ... Nachricht
N ... Modulus
$R = 2^m$ mit $m \in \mathbb{N}$, $R = 2^m > N$, $2^{m-1} < N$, $ggT(R, N) = 1$,
$1 \equiv R \cdot R^{-1} \mod N$

Ausgabe: $m = c^d \mod N$

1: $\overline{a} = 1 \cdot R \mod N$ → Initialisierung mit dem Wert Eins
2: $\overline{c} = c \cdot R \mod N$ → Transformation in den Montgomery-Raum
3: **for** $i = k - 1 \rightarrow 0$
4: $\quad \overline{a} = \overline{a} \circ \overline{a}$ → Quadrieren im Montgomery-Raum
5: $\quad$ **if** $d_i = 1$ **then**
6: $\quad\quad \overline{a} = \overline{a} \circ \overline{c}$ → Multiplizieren im Montgomery-Raum
7: $\quad$ **end if**
8: **end for**
9: $m = \overline{a} \circ 1$ → Rücktransformation in den Ursprungsraum

Die Montgomery-Multiplikation kann auch für die inverse Transformation, wie im Schritt 9 des Codes ersichtlich, verwendet werden. Dazu lädt man einen Operator mit dem Einselement aus dem Ursprungsraum und erhält folgende Vereinfachung:

$$m = \overline{a} \circ 1 = \overbrace{(a \cdot R \mod N)}^{\overline{a}} \cdot 1 \cdot R^{-1} \mod N = a \tag{3.32}$$

3.3 Anwendungen

Die Implementierung von Verschlüsselungsmethoden erfolgt immer öfters in einem „Embedded System“ in Consumer- oder Industrieelektronik, da mehr und mehr Geräte an das Internet oder

andere öffentliche Netzwerke angeschlossen sind. Dieser Trend ist unter dem Begriff „Internet der Dinge“ [Eva11] bekannt und stellt viele Entwickler vor neue Herausforderungen, wie die Umsetzung einer sicheren, optimierten Verschlüsselungsimplementierung für die Kommunikation.
In einem „Embedded System“ werden dazu oft FPGA-Strukturen eingesetzt, die durch ihre Parallelität Verschlüsselungen wie RSA energieeffizient in einem kurzen Zeitraum abarbeiten können [MMSS02].Die Anbindung eines FPGAs an einen Prozessor kann entweder mit einem externen FPGA oder direkt auf einem „System on Chip“ (SoC) erfolgen. Eine direkte Anbindung auf einem SoC hat den Vorteil, dass eine hohe Bandbreite für die Auslagerung der Verschlüsselung auf die FPGA-Struktur zur Verfügung steht [SSK10, S. 20-21].

Ein FPGA, der auf SRAM („Static Random-Access Memory“)-Technologie basiert, verliert seine Programmierung, wenn er nicht mit Spannung versorgt wird. Deswegen werden während des Einschaltvorgangs Daten für die Konfiguration aus einem externen Speicher geladen. Um sich vor einer Kopie der Logik-Programmierung zu schützen, bieten viele Hersteller von hardwareprogrammierbaren Strukturen die Möglichkeit an, diesen sogenannten „Bitstream“ verschlüsselt zu übertragen, was zu einem breiten Einsatz von verschiedenen Verschlüsselungsmethoden in FPGAs führt [4]. Die Authentifizierung des „Bitstreams“ kann dabei über die untersuchte RSA-Verschlüsselung erfolgen.

Public Key Verfahren, wie es die Algorithmen RSA und ECC[2] vorgeben, bieten dem Anwender die Möglichkeit, Nachrichten zu verschlüsseln und anhand von Zertifikaten zu überprüfen. Zertifikate helfen die Identität des öffentlichen Schlüssels der Übertragungspartner zu verifizieren. Außerdem werden sie für sichere Übertragungen von Webseiten eingesetzt und deshalb im Unterkapitel 3.3.1 im Detail beschrieben. Ein Einsatzgebiet für die RSA-Verschlüsselung findet sich in der digitalen Signatur, welche ebenfalls im Abschnitt 3.3.1 im Fokus steht. Das Teilkapitel 3.3.2 behandelt Protokolle und ihre Anwendungen, die auf den Verschlüsselungsalgorithmus RSA setzen.

3.3.1 Digitale Signatur und Zertifikate

Wie in [3] beschrieben, werden digitale Signaturen unter anderen für digitale Unterschriften - wie beispielsweise in der Finanzwelt über die Bürgerkarte - verwendet, um die Echtheit einer Nachricht zu überprüfen. Im Gegensatz zu einer herkömmlichen Verschlüsselung mit RSA, verwendet die digitale Signatur den privaten Schlüssel zum Signieren, da nicht die Geheimhaltung der Daten, sondern die Authentizität und die Integrität im Vordergrund stehen. Wenn sich die Nachricht nun mit dem öffentlichen Schlüssel entschlüsseln lässt, kann man davon ausgehen, dass diese vom Besitzer des geheimen Schlüssels signiert wurde.

Durch ein digitales Zertifikat in einer Public-Key-Infrastruktur kann ein Anwender einen öffentlichen Schlüssel seiner Identität zuordnen und dadurch auf seine Unverfälschtheit prüfen. Ein Zertifikat besteht in der Regel aber auch aus weiteren Inhalten, wie die Gültigkeitsdauer und dem Eigentümer des Datensatzes, sowie andere Attribute des Verschlüsselungsverfahrens [AL02, S. 69-70]. Um die Verteilung der Zertifizierungsdatensätze kümmert sich eine sogenannte „vertrauenswürdige Institution“ [Paw98, S. 11-12]. Digitale Signaturen, Email und der Aufruf von verschlüsselten Webseiten zählen dabei zu den Anwendungen digitaler Zertifikate.

[2] „Elliptic Curve Cryptography“

3.3.2 Verschlüsselungen in verteilten Netzen

Neben der besseren Skalierbarkeit und der optimierten Ressourcenausnutzung bieten verteilte Systeme den Vorteil, dass sie fehlertolerant arbeiten können. Eine Anforderung in verteilten vernetzten Systemen im „Embedded Systems“-Bereich ist meist, dass diese extern konfigurierbar sein müssen, um Änderungen einfach vornehmen zu können. Außerdem wird in dezentralen Netzen eine Kommunikation zwischen einzelner Knoten verlangt. Werden diese Daten in öffentlich zugänglichen Netzen unverschlüsselt übertragen, besteht die Gefahr, dass Systeme durch Manipulation und Sabotage von Dritten attackiert werden können.

Dieses Unterkapitel soll deshalb die wichtigsten verschlüsselten Kommunikationsprotokolle und die besondere Bedeutung von RSA in diesen Bereich hervorheben. Viele Applikationen verwenden für den sicheren Austausch eines symmetrischen Schlüssels die RSA-Methode, um anschließend eine symmetrische Verschlüsselung betreiben zu können. Diese Verschlüsselungskombination wird hybride Verschlüsselung genannt und wird aus Performancegründen eingesetzt, da symmetrische Verschlüsselungen für gewöhnlich um ein Vielfaches schneller sind.

Das „Transport Layer Security“ Protokoll, auch unter dem Akronym TLS bekannt, wird für die sichere Übertragung zwischen Client und Server im Internet eingesetzt, wie in [Die06] festgehalten. Zum Beispiel basiert das HTTPS-Protokoll[3] auf TLS beziehungsweise auf seiner Vorgängerversion „Secure Sockets Layer“, kurz SSL. Webserver, die TLS einsetzen, nutzen eine hybride Verschlüsselungsmethode, die einerseits eine asymmetrische Verschlüsselung, wie RSA, für den Schlüsselaustausch verwendet und andererseits eine symmetrische Verschlüsselungsmethode, die eine abhörrobuste Übertragung sicherstellt. Aber auch viele andere Applikationen verwenden diese fünfte Schicht („Session Layer“) im OSI-Modell. OSI steht für „Open Systems Interconnection“, was zu einem weiten Einsatz der RSA-Verschlüsselungsmethode führt.

Neben TLS, verwendet das sichere Remote-Login Protokoll „Secure Shell“ (SSH) ebenfalls dieses asymmetrische Verschlüsselungsverfahren für den Schlüsselaustausch [Har06].

3.4 RSA-ähnliche Verschlüsselungsverfahren

Dieses Kapitel stellt in den zwei Abschnitten 3.4.1 und 3.4.2 Verschlüsselungs- beziehungsweise Schlüsselaustauschmethoden vor, die große Ähnlichkeit zu den mathematischen Operationen in einer RSA-Verschlüsselung aufweisen. Das ist Grund dafür, dass sich die Interpretation der Seitenkanäle in einer möglichen FPGA-Implementierungen nicht besonders unterscheidet.

3.4.1 Diffie-Hellman Schlüsselaustausch

Ausgehend von der Publikation [DH76] beruht der Algorithmus für den Schlüsselaustauch von Deffie-Hellman darauf, dass aus den Werten $A = c^a \mod p$ und $B = c^b \mod p$ keine Rückschlüsse auf den geheimen Schlüssel $K = c^{a \cdot b} \mod p$ gezogen werden können. Die beiden öffentlichen Werte (c und p) müssen vor dem Schlüsselaustausch für beide Parteien bekannt sein, damit über die beiden geheimen Erzeugerschlüssel (a und b) die unverschlüsselt übertragenen Zwischenergebnisse (A und B) generierbar sind.

[3] HTTPS steht für „HyperText Transfer Protocol Secure“und kümmert sich um die verschlüsselte Übertragung von Webseiten

Nach dem Austausch der beiden Nachrichten führt nun auf beiden Seiten das Potenzieren der ausgetauschten Teilergebnisse mit dem eigenen geheimen Erzeugerschlüssel zum selben geheimen Schlüssel K.

Die RSA-Verschlüsselung zeigt eine starke mathematische Verwandtschaft mit dem Deffie-Hellman Schlüsselaustausch während der Berechnung der modularen Potenzen mit den Exponenten a, b sowie $a \cdot b$.

3.4.2 Elgamal Verschlüsselung

Als Grundlage für die folgende Beschreibung der Elgamal-Verschlüsselungsmethode wird in diesem Abschnitt die Arbeit [ElG85] verwendet. Angenommen Person A will Person B eine Nachricht m schicken, wobei $0 \leq m \leq p-1$ gilt. So muss Person A zuerst eine Zufallszahl k zwischen 0 und $p-1$ generieren. Anschließend berechnet diese den Schlüssel

$$K = (c^b)^k \mod p \tag{3.33}$$

aus dem öffentlichen Schlüssel $c^b \mod p$ der Person B. Für den nächsten finalen Schritt für die Verschlüsselung kommt der öffentliche Generator c zum Einsatz. Der übertragene, verschlüsselte Text besteht nun aus dem Paar

$$c_1 = c^k \mod p \tag{3.34}$$

$$c_2 = K \cdot m \mod p \tag{3.35}$$

Auf die ursprüngliche Nachricht kann nun der Empfänger B rückrechnen, da nur er seinen privaten Schlüssel b kennt und es nur ihm möglich ist, den Wert K aus c_1 zu errechnen.

$$\frac{c_2}{c_1^b} = \frac{m \cdot ((c^b)^k \mod p)}{(c^k \mod p)^b} = m \tag{3.36}$$

Im letzten Rechenschritt 3.36 spiegelt das modulare Potenzieren im Divisor die Mathematik von RSA wider.

4 Seitenkanalangriffsvektoren

Laut [BO08] können Angriffe auf Verschlüsselungsverfahren, welche Seitenkanalinformationen nutzen, unter dem Begriff Seitenkanalangriffe zusammengefasst werden. Typische Seitenkanäle entstehen beispielsweise durch zeitliche Effekte, variable Verlustleistung, elektromagnetische Vorgänge und akustische Geräusche im Zielsystem. Aktive Seitenkanalangriffe unterscheiden sich zu passiven dahingehend, dass der Angreifer auf das Verhalten des Zielobjekts einwirkt. Im Gegensatz dazu beobachten passive Angriffe das Verhalten ohne direkten Einfluss auf den Prozess (siehe Prinzipbild 4.1). Zu den passiven Attacken zählt man „Simple Power Analysis“ (SPA), „Differential Power Analysis“ (DPA), elektromagnetische Analyse (EMA) und Rechenzeitangriffe. Unter „Fault Analysis“ (FA) versteht man hingegen eine aktive Seitenkanalattacke.

Der bedeutende Unterschied zu anderen mathematischen Methoden, wie ein „Bruteforce“-Angriff, besteht darin, dass diese Art von Angriffsvektoren nicht, wie üblicherweise, stark durch die Erhöhung der Schlüssellänge erschwert wird, sondern sie mehr vom Umfeld und der Implementierung des Verschlüsselungsalgorithmus abhängt. Die Angriffsziele einer solchen Attacke reichen vom Personal Computer bis hin zu eingebetteten Systemen, wie Smartcards, anwendungsspezifische integrierte Schaltkreise (ASICs) und klassische Mikrocontroller. Die folgenden Unterkapitel stellen verschiedene Seitenkanalangriffsvarianten und dazugehörige Gegenmaßnahmen mit besonderem Fokus auf „Field Programmable Gate Arrays“ (FPGAs) vor.

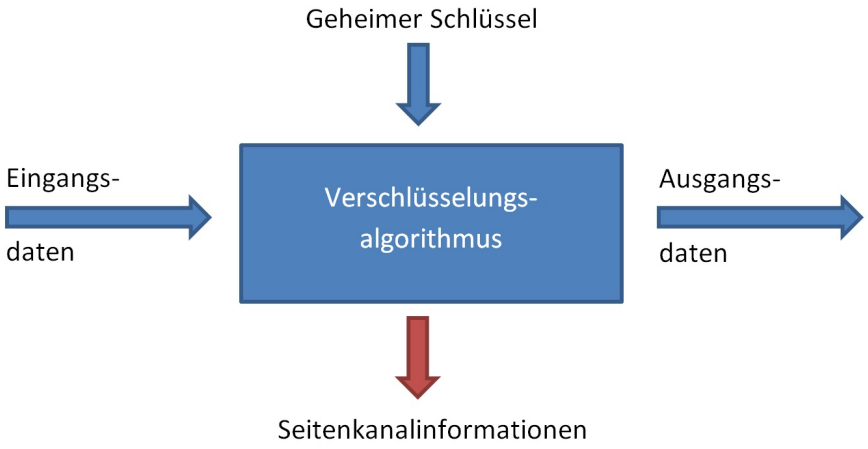

Abbildung 4.1: Prinzipbild eines passiven Seitenkanalangriffs

4.1 Simple Power Analysis

Eine häufig verwendete Methode für Seitenkanalangriffe nennt sich „Simple Power Analysis“ (SPA) und beruht auf der Messung des Stromverbrauchs der Komponente, die den Verschlüsselungsalgorithmus abarbeitet, über der Zeit. Die Variation des Stroms in der Abhängigkeit der auszuführenden Operationen ist technologieabhängig und stark mit der Umgebung, wie dem Rauschverhalten und gleichzeitig ablaufenden Tasks, verknüpft. Die Grundtechnologie der meisten integrierten Schaltkreise basiert auf der CMOS („Complementary Metal-Oxide-Semiconductor“)-Technik. Die Verlustleistung in dieser Technologie kann grundsätzlich in statische und dynamische Effekte eingeteilt werden:

- Die Veröffentlichung [Fis10, S. 9-10] beschreibt die Herkunft von dieser statischen, elektrischen Verlustleistung. Der größte Teil dabei wird durch Leckströme in den Transistoren, die die Grundlage eines CMOS bilden, verursacht. Dieser Strom entspricht dem Sperrstrom eines pn-Übergangs und ist technologie- und temperaturabhängig. Subschwellströme, die durch Ladungsträgerdiffusion im Transistor entstehen und Tunnelströme durch das Gate Oxid gelten als weitere Ursachen für den statischen Stromverbrauch.
  Prinzipiell gilt: Je kleiner die Strukturgröße beziehungsweise die Kanallänge, desto geringer der Widerstand und desto höher der damit verbundene statische Stromverbrauch. Mithilfe der statischen Analyse kann auf aktivierte oder deaktivierte Schaltungsteile rückgeschlossen werden.

- Kurzschlussströme treten während des Umschaltens eines CMOS auf und zählen deswegen zu den dynamischen Vorgängen. Sie werden im Detail in [Fis10, S. 11-12] beschrieben. Abbildung 4.2 zeigt dazu das Beispiel eines CMOS-Inverters, der aus einem NMOS („N-Channel Metal–Oxide–Semiconductor“)- und einem PMOS („P-Channel Metal–Oxide–Semiconductor“)-Feldeffekttransistor besteht. Die strichlierte Linie lässt den Stromfluss I_k erkennen, der dadurch entsteht, dass sich die beiden Transistoren für einen kurzen Zeitpunkt gleichzeitig im linearen Bereich befinden.

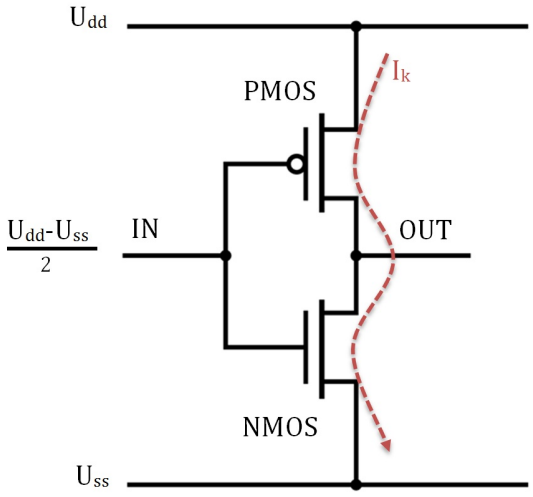

Abbildung 4.2: Verlauf des Kurzschlussstroms in einem CMOS Inverter [Fis10, S. 12]

Nach [Sta10] lässt sich ein anderer dynamischer Effekt durch das Laden und Entladen von Verbindungsleitungen und nachfolgenden Logikeinheiten in einem FPGA erklären. Diese

kapazitive Auswirkung kann über einen Lastkondensator C_L und die Verlustleitung

$$P = U_{dd}^2 \cdot C_L \cdot f_{CLK} \cdot \alpha \tag{4.1}$$

approximiert werden. Die Formel beinhaltet neben der Kapazität, auch die Versorgungsspannung U_{dd}, die Taktfrequenz f_{CLK} und die Aktivitätsrate α. Die Rate gibt die Wahrscheinlichkeit eines Schaltvorgangs an. Bei unterschiedlichen Signallaufzeiten in asynchronen Schaltteilen können zusätzliche ungewollte Signalwechsel beobachtet werden. Diese sogenannten „Glitches“ erhöhen die Aktivitätsrate und in weiterer Folge die dynamische Verlustleistung wesentlich.

Ein gutes Beispiel für eine mögliche Angriffsfläche für SPA wird im „Quadrieren und Multiplizieren“-Algorithmus (Algorithmus 3.1) ersichtlich. Abhängig vom privaten Schlüssel, werden entweder Daten quadriert oder multipliziert. Kann nun ein eindeutiges Muster der jeweiligen Operation im Stromverbrauch des zu untersuchenden Gerätes zugeordnet werden, so lässt sich aus diesem Wissen der Schlüssel rekonstruieren [MHAS08]. In einer Architektur mit Mikroprozessorbefehlen können bedingte Sprünge im Programmablauf verfolgt werden. Etwas erschwert wird diese Angriffsmethode in FPGAs, wo mehrere Abläufe parallelisiert in einem Takt abgearbeitet werden können. Die Voraussetzung für einen Erfolg einer solchen Attacke liegt vor allem im Signal-Rausch-Verhältnis des dynamischen Stromverbrauchs. Dieses Verhältnis kann durch „Differential Power Analysis“, welche im nächsten Unterkapitel erläutert wird, erhöht werden.

4.2 Differential Power Analysis

Wenn die Informationen über SPA für einen erfolgreichen Seitenkanalangriff nicht ausreichen, kann laut [KJJR11] „Differential Power Analysis“ (DPA) eingesetzt werden. Durch mehrere Messungen, Fehlerkorrektur und statistische Methoden lassen sich detaillierte Vorgänge im Zielsystem feststellen. Die Anzahl der Messungen hängt dabei stark vom Signal-Rausch-Verhältnis ab. Die Grundidee hinter DPA ist, dass zum selben Zeitpunkt der Programmausführung die Messung des Stromverbrauchs eines Geräts mit den Berechnungen statistisch korreliert.

Verschiedene statistische Methoden und Fehlerkorrekturmechanismen werden für diesen Zweck eingesetzt [Kre98, S. 160-162] [SLP05]:
In der Statistik spielen oft der Mittelwert und die Varianz von korrelierten Stichproben der Seitenkanalsignale eine Rolle. Folgende Schätzverfahren können dabei helfen, sinnvolle Ergebnisse aus den Messungen zu extrahieren.

- Maximum-Likelihood-Methode
- Methode der kleinsten Quadrate
- Minimum-Chi-Quadrat-Methode
- Momentenmethode

Die genaue Erklärung der einzelnen Verfahren ist für die weitere Vorgangsweise dieser Arbeit nicht von Relevanz und würde den Rahmen dieser Arbeit sprengen. Daher wird an dieser Stelle nicht weiter darauf eingegangen.

Je nach Vorgangsweise differenziert man im Weiteren zwischen verschiedenen Unterkategorien von DPA-Angriffen. Zu den drei am meisten verwendeten Methoden gehören „Correlation Power Analysis“ (CPA), „Template-based Analysis“ (TA) und „High-Order Differential Power Analysis“ (HO-DPA).

Correlation Power Analysis

Wie bereits erwähnt, kann je nach Implementierung, Messmethode und Umgebungseinfluss auf verschiedene Arten Rückschlüsse auf das Verhalten gezogen werden. Für die folgende vorgestellte Vorgehensweise aus [BCO04] muss zusätzlich der übergebene, verschlüsselte Klartext an die Verschlüsselungseinheit bekannt sein. Diese iterative Methode verlangt, dass die zuerst verarbeiteten Teile des gesuchten Schlüssels vorgegeben werden, woraus sich der Teilschlüssel stückweise aus den Ergebnissen der Korrelation zusammensetzt. Mit einem eigens erstellten Leistungsmodell gleicht man dazu, die aus dem Modell erzeugten Daten, mit den durch die Statistik aufgefassten Messungen ab und erhält somit die gewünschten Verhaltensinformationen. Ein solches Modell hängt meist von der Hamming-Distanz, also von dem binären Bitwechsel der gesuchten Variable a zum Vorgängerwert a', ab. Der Hamming-Abstand berechnet sich also in diesem Fall durch das Zählen der gesetzten Bits in $a \; XOR \; a'$.

In einem FPGA geben besonders Variablen in einer Prozessstruktur viele Informationen preis, wenn sie in einem Takt hintereinander, aber voneinander abhängig, geändert werden. Dieser Umstand ist auf temporäre logische Falschaussagen in Folge eines Bitwechsels in einem FPGA rückzuführen und wurde bereits in Unterkapitel 4.1 unter dem Fachbegriff „Glitch“ erläutert.

Eine mögliche Vorgehensweise für diesen Angriff sieht wie folgt aus:

1. Aufzeichnen des Seitenkanals und der Eingangsdaten in die Entschlüsselungseinheit über mehrere Messungen
2. Vorverarbeiten der Messwerte über Filter und Synchronisieren der Messzeitpunkte
3. Mitteln der Messungen mit denselben Eingangsdaten
4. Vorhersagen des Teilschlüssels
5. Generieren von Simulationsdaten für die Schlüsselvorhersage anhand der Eingangsdaten über ein Verlustleistungsmodell
6. Abgleichen der Messungen mit der Simulation (Korrelation)
7. Abhängig von der Übereinstimmung mit der Vorhersage des Teilschlüssels diesen ändern oder erweitern

   Die Iteration der Schritte Vier bis Sieben wird solange wiederholt, bis der geheime Schlüssel gefunden wurde.

Für den Abgleich der Simulationsdaten mit dem Modell stehen mehrere mathematische Konstrukte zur Verfügung. Das am häufigsten verwendete Konstrukt für die Korrelation in der Statistik nennt sich Produkt-Moment-Korrelation [Hei10, S. 147-148] und ist auch unter dem Namen Pearson-Korrelationskoeffizient bekannt. Dieser Faktor ist normalisiert - weswegen er sich immer

zwischen den Werten Eins und minus Eins befindet - und misst den linearen Zusammenhang zwischen zwei Signalen in einem Intervall.

Der Korrelationskoeffizient berechnet sich aus der Kovarianz des Modells M mit der Messung X dividiert durch das Produkt der beiden Standardabweichungen. Daraus ergibt sich die Formel für den Koeffizienten zu [BCO04]:

$$\rho(X,M) = \frac{N \cdot \left[\sum_{i=1}^{N}\left(X_i \cdot M_i\right)\right] - \left[\left(\sum_{i=1}^{N} X_i\right) \cdot \left(\sum_{i=1}^{N} M_i\right)\right]}{\sqrt{\left[N \cdot \left(\sum_{i=1}^{N} X_i^2\right) - \left(\sum_{i=1}^{N} X_i\right)^2\right] \cdot \left[N \cdot \left(\sum_{i=1}^{N} M_i^2\right) - \left(\sum_{i=1}^{N} M_i\right)^2\right]}} \tag{4.2}$$

Der Index i identifiziert den aktuellen Messzeitpunkt der Übereinstimmung und die Konstante N bestimmt die gesamte Anzahl der Punkte im Vergleichsintervall. Befindet sich der Koeffizient $\rho(X,M)$ in der Nähe seines positiven Maximalwerts, so spricht man von einem positiven linearen Zusammenhang, was auf eine Übereinstimmung der Annahme mit internen Vorgängen hinweist.

Template-based Analysis

Eine Template Attacke (TA) unterteilt sich nach [CRR03] in zwei Phasen. In der ersten Phase werden zuerst Templates in einem identischen Device oder bestenfalls im Zielsystem für alle möglichen Operationen generiert. Diese Templates beinhalten einerseits Informationen über das typische Signal selbst und andererseits auch Parameter, die das Rauschen beschreiben. Während der zweiten Phase ordnet der Angreifer die gemessene Verlustleistung der gesuchten Operation am Zielsystem einem Template zu. Diese Zuordnung übernimmt für gewöhnlich die „Maximum-Likelihood"-Methode. Das Rauschen wird dabei als normalverteilt angenommen. Diese Attacke zählt zu den effektivsten Methoden, da sie nur wenige Messungen benötigt und daher schnell und robust ist. Jedoch muss dafür mindestens ein Device mit den dazugehörigen Programmierfiles zur Verfügung stehen, die hohe Ähnlichkeit mit denen des Zielsystems haben. Diese Anforderung ist für gewöhnlich für einen Blackbox-Angriff nicht erfüllt und kommt deswegen für diesen Zweck auch nicht zum Einsatz.

In Abbildung 4.3 sieht man links die Differenz des Spannungseinbruchs an einem FPGA zwischen einer Operationen mit dem dazugehörigen Template. Das rechte Teilbild beinhaltet den Unterschied zwischen demselben Template zu einer anderen Operation, die erst in der Mitte der Messaufnahme vom verglichenen Template in der Ausführung abweicht. Der durchgehende, minimale Ausschlag der Differenz im linken Bild bestätigt die Übereinstimmung mit dem Template, welches für diesen Versuch aus demselben Gerät generiert wurde.

High-Order Differential Power Analysis

„High-Order Differential Power Analysis", kurz HO-DPA, ist eine erweiterte Form von DPA. Der Angriff verwendet nicht nur verschiedene Quellen für die Analyse, er kommt auch mit verschiedenen zeitlichen Offsets zurecht [GBPV10]. Im Vergleich zu SPA und DPA kommt dieser komplexe Angriffsvektor nur selten zum Einsatz. Die Verwendung von HO-DPA lohnt sich erst dann, wenn Implementierungen durch Gegenmaßnahmen geschützt sind [Mes01].

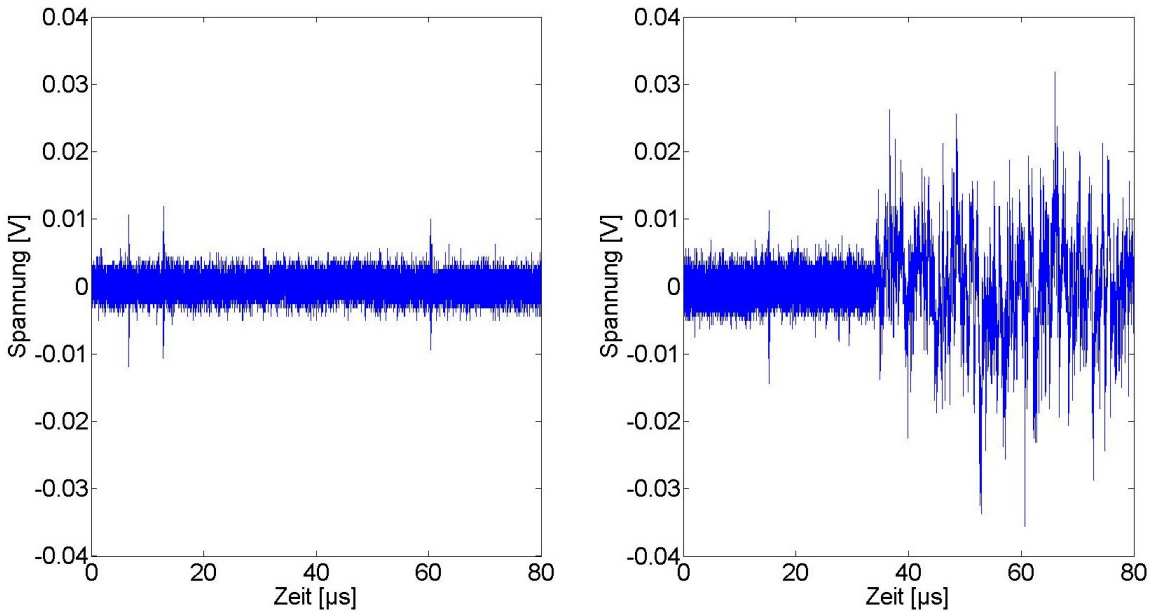

Abbildung 4.3: Template Angriff - Links: Differenz mit richtigem Template - Rechts: Differenz mit falschem Template

4.3 Rechenzeitanalyse

Unter dem Begriff „Timing Attack“ oder Rechenzeitanalyse versteht man eine Seitenkanalattacke, in der der Angreifer aufgrund der Ausführungszeit von Algorithmen internes Verhalten beobachten kann [Koc96]. Dabei muss der Algorithmus, abhängig von seinen Eingangsdaten oder Schlüsseldaten, unterschiedlich lange Berechnungszeiten aufweisen. Das Ende seiner Ausführung kann zum Beispiel durch eine Benachrichtigung des Systems, wie es in Grafik 4.4 sichtbar ist, oder durch das Messen seiner momentanen Leistung, wenn diese eindeutig das Ende der Ausführung kennzeichnet, erkannt werden.

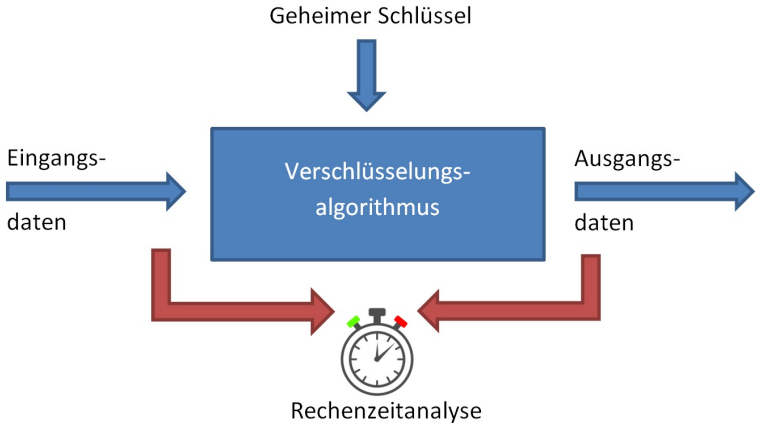

Abbildung 4.4: Prinzipbild einer Rechenzeitanalyse

Nach [Koc96] können besonders bei asymmetrischen Verschlüsselungen, wie RSA, unterschiedliche Ausführungszeiten beobachtet werden. Diese Variation ist stark von der Implementierung abhängig. Das binäre modulare Potenzieren, welches im Zuge dieser Arbeit analysiert wird, lässt auf das Hamming-Gewicht des privaten Schlüssels rückschließen. Je mehr gesetzte Bits der private Schlüssel aufweist, desto länger benötigt der Algorithmus für die Entschlüsselung der Information. Dieser Algorithmus arbeitet zeitlich unabhängig von den zu entschlüsselnden Daten.

Im Falle der Umsetzung des Algorithmus auf Grundlage des chinesischen Restsatzes - Informationen dazu im Kapitel 3.2.1 - kann es jedoch durch veränderte Eingangsdaten bei unverändertem Schlüssel zu einer unterschiedlichen Ausführungszeit kommen [Koc96]. Aus diesem Grund spielt dieser Angriffsvektor für das Detektieren von Verschlüsselungsalgorithmen in Blackbox Systemen eine wichtige Rolle.

4.4 Electro-Magnetic Analysis

Unter dem englischen Begriff „Electro-Magnetic Analysis“ versteht die Arbeit [AARR03] den elektromagnetischen Seitenkanalangriff. In dieser Analyse wird die elektromagnetische Abstrahlung des Zielobjekts gemessen und anschließend ausgewertet. Ähnlich wie bei den Angriffsvektoren SPA und DPA können so interne Vorgänge beobachtet werden. Ein über der Logikeinheit platzierter elektromagnetischer Sensor misst dabei die durch innere Vorgänge hervorgerufene, impulsartige Stromflussänderung im Inneren des Chips. Der Sensor kann, abhängig von seiner Winkelausrichtung und Position, verschiedene Messwerte zum gleichen Ausführzeitpunkt aufnehmen. Diesem Umstand folgt eine aufwendigere Messung bei hoher Granularität und eine komplizierte Auswertung. Solange kein hardwaretechnischer Schutz gegen das Messen der Abstrahlung vorhanden ist, besteht der Messaufbau nur darin, den Sensor möglichst nahe an das Objekt zu führen.

4.5 Fault Analysis

Hinter der Abkürzung FA versteckt sich die Methode der „Fault Analysis“ [BDL97]. Es handelt sich dabei um eine besondere Form von Seitenkanalangriffen, in welcher gezielt Fehler über Seitenkanäle in das Analyseobjekt induziert werden. Unter Fehlern versteht man hier gezielte Manipulation von einzelnen Bits oder mehreren Bytes, die ausgehend von einem mathematischen Modell vorgegeben werden. Folgende Aufzählung listet verschiedene Möglichkeiten diese Fehler zu generieren auf [Zef07, S. 10-12]:

- Laser
- fokussierter Ionenstrahl
- elektromagnetische Felder
- lokale Temperaturerhöhung
- unregelmäßige Versorgungsspannung
- Taktsignalstörung

Diese fehlerverursachenden Methoden können zur Beschädigung - bis hin zur Zerstörung - des Geräts führen. Die Fehlerinduzierung findet jedoch immer häufiger Anwendung, da diese neue effektive Angriffsmöglichkeiten schafft.

4.6 Gegenmaßnahmen

In diesem Kapitel werden verschiedene Maßnahmen zur Verteidigung von Seitenkanalangriffen vorgestellt. Einzelne Abwehrmechanismen verteidigen meist nur bestimmte Typen von Angriffen und sollten aus diesem Grund in Kombination verwendet werden [GM11]. Es gilt der einfache Grundsatz: Je mehr Schutzmechanismen eingesetzt werden, desto geringer ist die Wahrscheinlichkeit Opfer eines erfolgreichen Angriffs zu werden. Die Klassifikation in algorithmische und hardwaretechnische Methoden hilft, diese Maßnahmen grob einzuteilen. Ein besonderer Fokus liegt dabei auf den Angriffsmethoden SPA, DPA, FA, EMA sowie der Rechenzeitanalyse.

Algorithmische Gegenmaßnahmen

Nach [Zef07, S. 8-9] basieren softwaretechnische Schutzmaßnahmen, die auf Angriffe wie SPA, DPA und EMA abzielen, typischerweise entweder auf Maskieren oder Verstecken von messbaren Informationen und erhöhen lediglich den Aufwand eines Angriffs. Die Methode des Maskierens baut dabei auf einem generierten Maskenwert auf, der die Verarbeitung der Daten oder des Schlüssels randomisiert und somit aus dem Zusammenhang mit dem Informationsmodell bringt. Dies geschieht entweder durch neutrale Parameteränderungen zu Beginn der Verschlüsselung, die sich nicht auf das Endergebnis auswirken [Bau12] oder durch die Manipulationen der Daten vor und nach dem Verschlüsselungsvorgang [BK08].

Das Verstecken von Seitenkanälen beschreibt [Zef07, S. 8-9] und wird dadurch gelöst, dass das SNR des Kanales reduziert wird. Dies kann durch parallel aktive Logik passieren, wie es in einem FPGA möglich ist, die nicht direkt mit der Verschlüsselung im Zusammenhang steht und daher das Rauschen verstärkt. Eine Verringerung der relevanten, variablen Spannungsamplitude lässt sich hingegen gut durch die Aufteilung der Aufgaben in eine größere Anzahl an Taktzyklen realisieren, da dadurch weniger Logik gleichzeitig beansprucht wird. Die ersten Messversuche in Kapitel 7 spiegeln diesen Sachverhalt wider. Effektive Algorithmen, die auf die Montgomery-Multiplikation oder dem chinesischen Restsatz Theorem aufbauen, können auch dazu beitragen, mit geringerer Logik auszukommen. Gegen die statistischen Methoden, welche im Zuge eines DPA-Angriffs vorkommen, helfen bei geringem SNR randomisierte Pausen und nicht im Zusammenhang stehende, zufällige Zwischenoperationen mit einem ähnlichen Leistungsprofil.

Um einen Angriff über den aktiven Seitenkanal „Fault Analysis“ zu erkennen, gibt es mehrere Varianten, welche in [Zef07, S. 11] erläutert sind. Einerseits können redundante Berechnungen externe Manipulationen detektieren, indem sie miteinander verglichen werden. Zum anderen wäre es sinnvoll, nach unerwarteten Zuständen beziehungsweise Unregelmäßigkeiten, welche durch solch einen Angriff entstehen können, zu filtern. Eine andere Methode initialisiert einige Speicherzellen und lässt diese unverändert gespeichert. Ein ständiger Abgleich auf Änderungen von außerhalb hilft, aktive Zugriffe wahrzunehmen. Nach einer erfolgreichen Detektion von externen Manipulationen kann es zum Abbrechen der Ausführung bis hin zur permanenten Sperre der Funktion kommen.

Gegenmaßnahmen, welche die Ausführungszeiten konstant halten und somit eine Resistenz gegen Rechenzeitangriffe aufweisen, gehören zu der Klasse des Versteckens von Seitenkanälen. Ein sehr bekanntes Beispiel dazu nennt sich „Montgomery Power Ladder“ [JY03] und kann dem Algorithmus 4.1 entnommen werden. Dieser hat große Ähnlichkeiten mit dem bereits vorgestellten binären modularen Potenzieren und balanciert die beiden darauf basierenden Operationen zeitlich aus. Die Ausführungszeit wird dadurch auf die maximal mögliche Ausführungszeit erweitert.

Algorithmus 4.1 : Montgomery Power Ladder

Eingabe:
k = Binärer geheimer Schlüssel
g = Nachricht
Ausgabe: $y = g^k$

1: $R_0 = 1$ → Initialisieren
2: $R_1 = g$
3: **for** $j = t - 1 \rightarrow 0$
4: **if** $k_j = 0$ **then**
5: $R_1 = R_0 \cdot R_1$
6: $R_0 = R_0 \cdot R_0$ → Quadrieren
7: **else**
8: $R_0 = R_0 \cdot R_1$ → Multiplizieren und
9: $R_1 = R_1 \cdot R_1$ → Quadrieren
10: **end if**
11: **end for**
12: $y = R_0$

Der große Nachteil dieser Gegenmaßnahmen besteht vor allem darin, dass daraus zusätzlich Logikverbrauch und Aufwand entsteht. Dies führt wiederum zu einer schlechteren Gesamtperformance.

Hardwaretechnische Abwehrmechanismen

Neben modifizierten Algorithmen verkomplizieren Gegenmaßnahmen, die in Hardware umgesetzt sind, ebenfalls Seitenkanalangriffe. Entweder geschieht dies durch einen Schaltungsaufbau, der die Messaufnahmen der Messobjekte nur schwer zugänglich macht oder durch eine andere hardwaretechnische Maßnahme, die das Messsignal versteckt, wie in [Man04] vorgestellt. Ein Großteil dieser Maßnahmen muss bereits während der Planung der Schaltung berücksichtigt werden, da sie nachträglich nur teuer hinzugefügt werden können. Ein absichtlich generiertes Jitter im Taktsignal kann beispielsweise vor Rechenzeitanalysen schützen und vor allem vor statistischen Methoden bewahren. Mehrere Messaufnahmen, die dasselbe Verhalten zeigen, werden so nicht mehr direkt zeitlich korrelieren. Ein erhöhtes Rauschen durch Rauschgeneratoren oder eine möglichst niedrige Versorgungsspannung kann wie auf algorithmischer Ebene ein niedrigeres SNR im Seitenkanal erzeugen.

Eine elektromagnetische Schirmung oder die Verwendung eines Schaltungsdesigns, in dem Messpunkte für Seitenkanalangriffe nicht direkt abgreifbar sind, erschweren zusätzlich den möglichen Messaufbau.

Eine Übersicht der wichtigsten vorgestellten Gegenmaßnahmen verschafft das Blockdiagramm in Abbildung 4.5.

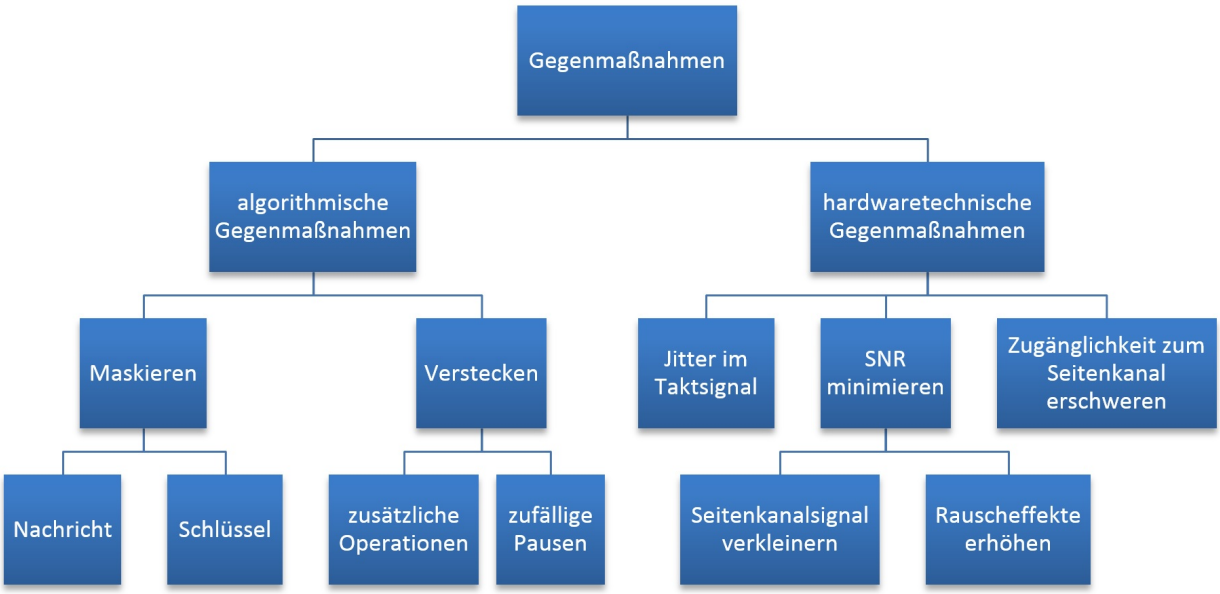

Abbildung 4.5: Übersicht der vorgestellten Gegenmaßnahmen

5 Methoden zur Detektion von ausgewählten Algorithmen

In diesem Kapitel werden Methoden vorgestellt, die versuchen, den Algorithmus einer FPGA-Implementierung zu identifizieren. Die Detektion des Algorithmus mit seinen Eigenschaften stellt sich als Vorbedingung für einen möglichen nachfolgenden Angriff heraus, da der Angreifer es meist mit einer Art Blackbox zu tun hat, deren genauer Inhalt vom Entwickler des Gerätes nicht zur Verfügung gestellt wird. Diese Detektionsmethoden leiten sich teilweise von existierenden Seitenkanalangriffen ab und ermöglichen einem Angreifer die Erkennung der groben Implementierung des Verschlüsselungsvorgangs. Erst nach dem Sammeln dieser Informationen, ist es sinnvoll, den eigentlichen Angriff zu optimieren und die Seitenkanalanalyse auf den vorliegenden Fall anzupassen.

Einen Überblick über die, in dieser Diplomarbeit erarbeiteten, Detektionsmethoden und ihren Zusammenhang in einer Blackbox-Analyse gibt der erste Unterabschnitt 5.1. Kapitel 5.2 beschäftigt sich mit der Bitbreite der Verschlüsselung und den damit verbundenen Auswirkungen. Im anschließenden Kapitel 5.3 werden Detektionsmethoden präsentiert, die auf der Auswirkung von variablen Eingangsdaten beruhen. Die Reduzierung der Eingangsdaten zu Beginn des Algorithmus mit dem chinesischen Restsatz Theorem und die Detektion von datenunabhängigen Operationen stehen hier im Vordergrund. Der Detektionsvorgang, der als Basis spezielle Eingangsdaten verwendet, wird im Teilabschnitt 5.4 behandelt. Dieser Abschnitt zeigt, dass Extremwerte nur für bestimmte Algorithmen einen charakteristischen Messausschlag verursachen. Andere Anhaltspunkte gibt die Abschnittsanalyse (Kapitel 5.5), die durch die Identifikation von unterschiedlichen Operationen in der Verlustleistungsmessung Algorithmen ausfindig macht. Der Einsatz eines Verlustleistungssimulationsmodells für die Detektion wird in Kapitel 5.6 behandelt.

Neben der Vorstellung der erarbeiteten Detektionsmethoden, betrachten diese Unterabschnitte auch ihre mögliche Anwendung auf die vier ausgewählten Algorithmen:

- Blakley-Multiplikation
- Montgomery-Multiplikation
- Binäres modulares Potenzieren
- Chinesisches Restsatz Theorem

Welche Ansätze sich prinzipiell für das Erkennen von Algorithmen über Seitenkanäle eignen oder welche eher ungeeignet dafür scheinen, behandelt Unterkapitel 5.7. Im Anschluss hilft die Abbildung 5.5 einen Überblick über die in diesem Kapitel besprochenen Analysewerkzeuge zu schaffen. Maßnahmen, die ein Entwickler treffen kann, um die Implementierung vor solchen Detektionsmethoden zu schützen, erläutert Abschnitt 5.8.

5.1 Angriffsmethodik

Am Beginn einer Blackbox-Analyse über Seitenkanäle steht immer die zu untersuchende Blackbox, für die ein funktionierender Seitenkanalmessaufbau erforderlich ist. Die vorgestellten Seitenkanaldetektionsmethoden gehen von einer solchen existierenden Messschaltung für die Verlustleistung des untersuchten FPGAs aus. Der Programmablaufplan in Grafik 5.1 bildet dazu eine systematische Vorgangsweise ab, die sich in dieser Arbeit als sinnvoll herausgestellt hat.

Die ersten eingesetzten Operationen für die Blackbox-Analyse verwenden nur Ergebnisse aus Beobachtungen und greifen nicht direkt in den Prozess ein. Kann aus diesen Beobachtungen nicht ausreichend Information für eine Identifikation gewonnen werden, muss der Angreifer im nächsten Schritt die Eingangsdaten für die untersuchte Verschlüsselung vorgeben können. Diese Angriffsvektoren zählen zu den effektivsten Detektionsmethoden, da sie oft ein markantes detektierbares Muster verursachen. Die letzte mögliche Detektionsoperation nutzt Simulationsmodelle für die Verlustleistung. Dazu werden Detailinformationen für die Detektion oder eine Datenbank mit verschiedenen Simulationsmodellen für jeden Algorithmus, wie in Abschnitt 5.6 genauer beschrieben, benötigt.

Führen diese Analysen zu keiner Identifikation des Algorithmus, kann eine Überarbeitung der Messschaltung helfen, das Seitenkanalsignal zu verbessern. Bei erfolgreicher Detektion muss im Weiteren die Seitenkanalanalyse der vorliegenden Implementierung angepasst werden, damit der Schlüssel aus dem Seitenkanal extrahiert werden kann.

Die im Programmablaufplan vorkommenden Operationen werden in den weiteren Abschnitten genauer vorgestellt.

5.2 Untersuchung und Auswirkung der Bitlänge

Die Bitlänge der Verschlüsselung hat für gewöhnlich direkten Einfluss auf die Dauer der Ausführung von Verschlüsselungsalgorithmen. Wenn der Entwickler keine Auskunft über diese Länge gibt und die geladenen Nachrichten, sowie der öffentliche Schlüssel nicht beobachtbar sind, kann der Angreifer durch Abschätzung des zeitlichen Aufwands versuchen, die Verschlüsselungsstärke zu eruieren. Für eine n Bit Implementierung benötigt der „Quadrieren und Multiplizieren“-Algorithmus zwischen n und $2n$ Operationen, die das temporäre Ergebnis entweder mit dem verschlüsselten Wert oder mit sich selbst multiplizieren. Dazu muss es in der Analyse möglich sein, verschiedene Operationen zu identifizieren.

Eine Implementierung, die auf dem chinesische Restsatz Theorem (CRT-RSA) beruht, berechnet die Verschlüsselung aufgrund ihrer Unterteilung in zwei kleinere Exponentialfunktionen schneller als das binäre modulare Potenzieren. In Summe ergeben sich dadurch zwar wieder n bis $2n$ Operationen, diese sind jedoch aufgrund ihrer halbierten Bitbreite deutlich effizienter realisierbar

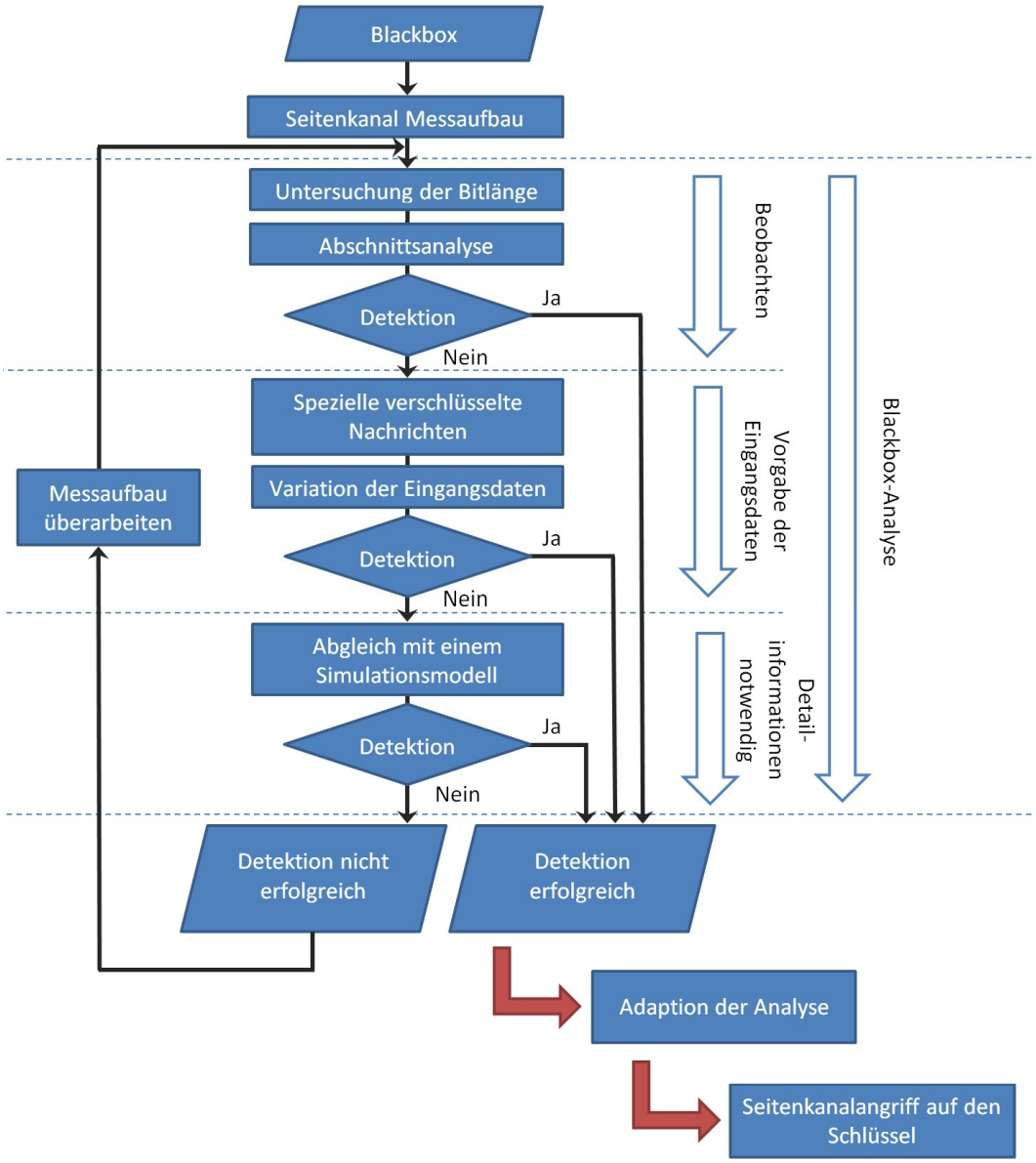

Abbildung 5.1: Blackbox-Angriffsmethodik

und in einem kürzeren Zeitraum abgearbeitet. Dieser mögliche Zeitraum ist stark abhängig von der Implementierung, was dazu führt, dass diese Laufzeitanalyse nur bedingt für die Detektion einsetzbar ist.

5.3 Variation der Eingangsdaten

Die Unterscheidung zwischen verschiedenen Algorithmen lässt sich in dieser Methode über die Variation der angelegten, zu entschlüsselten Nachricht treffen. Hat ein Algorithmus abhängig von den Eingangsdaten unterschiedliche Ausführungszeiten, so können Rückschlüsse auf die Implementierung gezogen werden.

Die Variation der Eingangsdaten kann auch für andere Zwecke eingesetzt werden - zum Beispiel für das Finden von Berechnungen, die nichts mit dem eigentlichen Entschlüsselungsvorgang zu tun haben und deswegen im Seitenkanal zum identischen Zeitpunkt dieselbe Form annehmen. Das könnten primitive, künstlich eingefügte Operationen sein, die einen Angriff erschweren sollen. Es kann sich aber auch um vom Schlüssel abhängige Vorberechnungen handeln, die für anschließende Operationen benötigt werden. Ändern sich weite Teile des Seitenkanalsignals bei Verschlüsselungsvorgängen mit gleichen Eingangsdaten, kann dies einerseits an starken Störungen im Messkanal liegen und andererseits auch an den Gegenmaßnahmen, die die Verschlüsselung vor dem Verschlüsselungsvorgang randomisieren.

Zum Beispiel kann die notwendige Vorberechnung im CRT-RSA-Algorithmus helfen, sich zu einer üblichen Potenzierfunktion abzugrenzen. Die Modulo-Operation, die immer am Beginn einer Verschlüsselung mit CRT-RSA steht, bestimmt dieses Verhalten. Durch die Vorgabe einer kleinen beziehungsweise einer großen Zahl, kann der Angreifer ohne großen Aufwand die Modulo-Operation im momentanen Leistungsprofil zeitlich lokalisieren und detektieren.

Die zwei Teilbilder in Abbildung 5.2 verdeutlichen diese Detektionsmethode, wobei der Spannungseinbruch - der negativ proportional zur Stromverbrauchsänderung im FPGA ist - auf der Ordinatenachse aufgetragen ist. Die untere Messung zeigt das Profil eines kleinen Werts und die obere den entgegengesetzten Fall. Dieses Verhalten findet sich in vielen nicht seitenkanalgeschützen Umsetzungen einer Modulo-Operation wieder. Bei der datenunabhängigen Vorberechnung in der Grafik handelt es sich um zwei Modulo-Berechnungen auf den geheimen Schlüssel, weshalb der Spannungseinbruch ein ähnliches Muster, wie die Reduktion großer Eingangsdaten, aufweist.

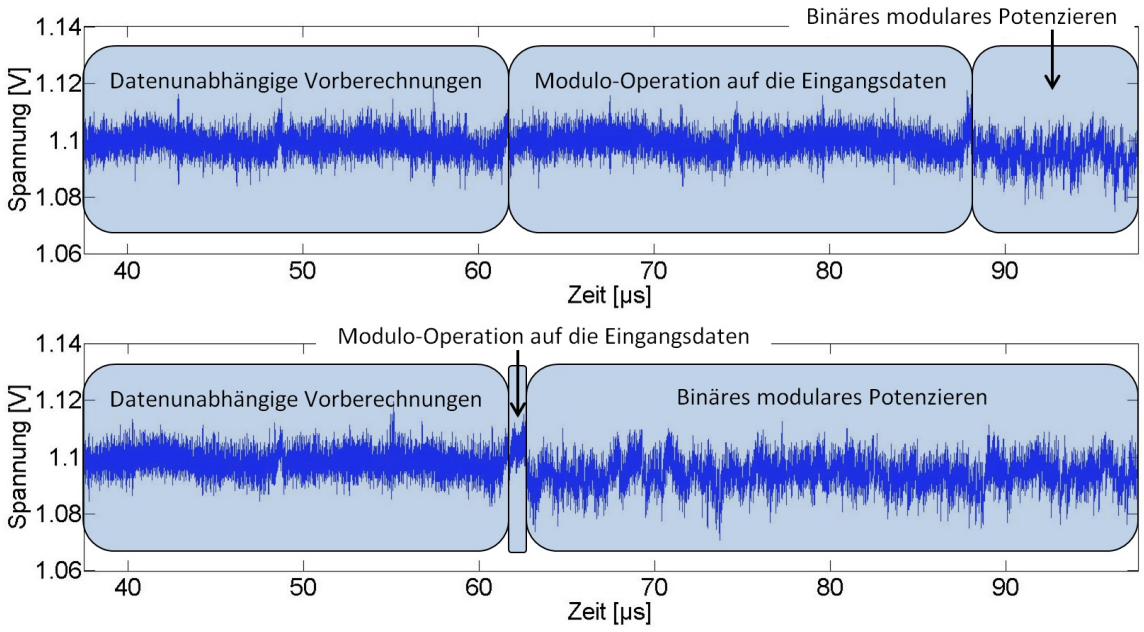

Abbildung 5.2: Detektion der Modulo-Operation über die Rechenzeit - Unten: Kleiner Eingangswert - Oben: Großer Eingangswert

Diese deutliche Änderung in der Ausführung kann auch anhand der unterschiedlichen Gesamtausführungszeit ohne dem Leistungsprofil gemessen werden.

5.4 Analyse spezieller verschlüsselter Nachrichten

Dieser Analysevorgang basiert auf der Tatsache, dass spezielle Nachrichten ein detektierbares Muster in der Verlustleistung verursachen. Die Messung eines herkömmlichen Verschlüsselungsvorgangs verrät ohne zusätzliche Informationen nur wenig über die Unterschiede zwischen den vom Schlüssel abhängigen Rechenvorgängen. Aufgrund dessen, dass verschiedene Algorithmen unterschiedliche Zwischenergebnisse berechnen, entsteht mit dieser Methodik eine effektive Unterscheidungsmöglichkeit anhand bestimmter, algorithmenspezifischer Nachrichten.

Ziel dieser Methode ist es, einerseits das binäre modulare Potenzieren im Seitenkanal der Blackbox ausfindig zu machen und andererseits die Art der Implementierung dieser Operation zu analysieren. Für diese Problemstellung finden Extremwerte Anwendung, wie sie häufig in dem Angriffsvektor SPA verwendet werden, um einen Schlüssel zu extrahieren [MHAS08]. Im Fokus dieses Unterkapitels steht jedoch mehr die Detektionsmethodik von Algorithmen und ihre Unterschiede, als die Extraktion des Schlüssels.

Setzt man kleine verschlüsselte Werte als Eingangsdaten für ein unbekanntes System ein, so lässt sich die Blakley-Multiplikation im Leistungsprofil durch besonders aktive und inaktive Bereiche im Spannungseinbruch demaskieren. Abbildung 5.3 zeigt diese aufgenommenen Messeffekte. Der erhöhte Verlustleistungsverbrauch - im Bild ersichtlich als der Abschnitt mit einem starken Spannungseinbruch - kann dem Quadrieren einer großen Zahlen zugeschrieben werden. Hingegen weist der Bereich mit geringerem Spannungseinbruch auf die Multiplikation einer großen Zahl mit dem kleinen vorgegebenen Extremwert hin.

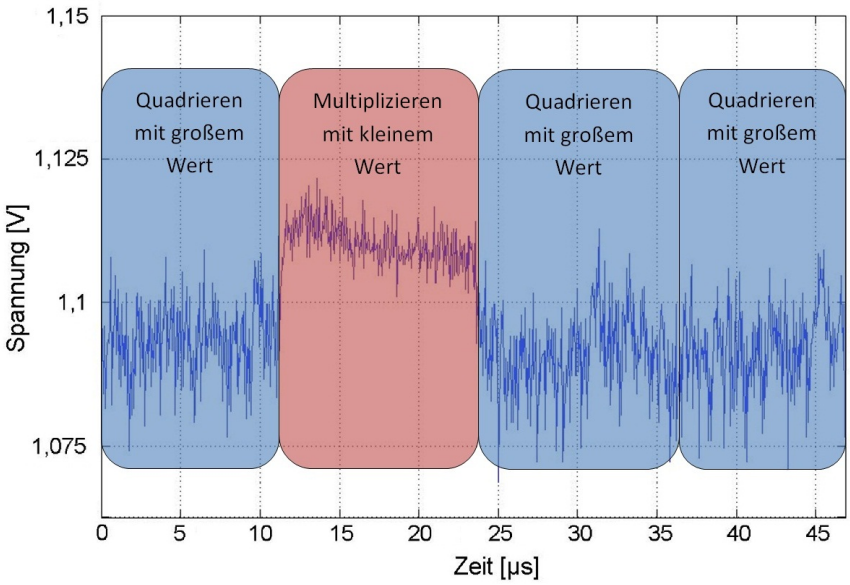

Abbildung 5.3: Blakley-Detektion mit einem kleinen Extremwert

Finden sich mit dieser Methode keine deutlichen Differenzen zwischen Quadrieren und Multiplizieren, so kann eine kleine Zahl X im Montgomery-Raum helfen, die Montgomery-Multiplikation sichtbar zu machen. Um diese Zahl im Ursprungsraum zu finden, muss der Transformationswert R, der sich aus dem öffentlichen Modulus N ergibt, die kleine Zahl aus dem Montgomery-Raum rücktransformieren. Für eine n Bit Verschlüsselung, mit der Voraussetzung, dass das

höchstwertigste Bit im Modulus gesetzt ist, liegt dieser Wert bei 2^n. Die Zahl C im Ursprungsraum, welche für den Angriff verwendet wird, berechnet sich aus der Montgomery-Operation zu $C = X \circ 1 = X \cdot R^{-1} \mod N$.

Mit der Vorgabe des Extremwerts $N-1$, wobei N in der RSA-Verschlüsselungsmethode öffentlich bekannt ist, steht einem Analysten ebenfalls eine Methode zur Verfügung, die Blakley- und Montgomery-Multiplikation in einem „Quadrieren und Multiplizieren"-Algorithmus zu finden [SSH+08]. Für diese Vorgehensweise muss es möglich sein, drei verschiedene, immer wiederkehrende Abschnitte in der Seitenkanalmessung zu detektieren. Diese drei Abschnitte stehen für die Multiplikation des binären Werts 1 mit $N-1$, das Quadrieren der Zahl 1 sowie $N-1$ im Ursprungsraum oder für die transformierten Operationen im Montgomery-Raum.

Im Vergleich zu einer Montgomery-Multiplikation zeichnet sich eine Blakley-Multiplikation dadurch aus, dass zwei von drei Multiplikationsoperationen einen geringen Stromverbrauch aufweisen. Im transformierten Raum hingegen nehmen die drei Operationen in der Leistungsmessung - abhängig vom Transformationsvektor und dem verwendeten Extremwert - einen transformierten Pseudozufallsmessausschlag an. Eine daraus resultierende zufällige Ähnlichkeit zwischen diesen Operationen kann dazu führen, dass eine Unterscheidung bei geringem Signal-Rausch-Verhältnis nur bedingt möglich ist. Verfügt der Angreifer über ein genaueres Leistungssimulationsmodell, so kann dieser die drei Operationen im Messsignal vorbestimmen und leichter detektieren.

Ein großer Vorteil von Extremwerten besteht darin, dass sie nicht stark von der Variation der Implementierung abhängen und auf algorithmenspezifische Zwischenwerte bauen. Deshalb zählen sie zu den effektivsten Seitenkanalangriffen für Detektionsmethoden.

5.5 Abschnittsanalyse

Diese Methode versucht verschiedene Muster im Seitenkanal während der Ausführung der Verschlüsselung zu lokalisieren. Betrachtet man das Leistungsprofil eines Verschlüsselungsvorgangs auf einem FPGA im Vergleich zu Nicht-Verschlüsselungsaufgaben, so lässt sich seine Aktivität durch die Suche von starken Variationen in der Leistungsmessung identifizieren. Aber auch während des Berechnungsvorgangs kann es - abhängig von der Implementierung - zu Unterschieden kommen. Verschiedene, mit dem freien Auge oder über statistische Methoden sichtbare Abschnitte beruhen auf unterschiedlichen Rechenoperationen, die je nach Algorithmus angewandt werden.

Der Algorithmus, der auf dem chinesischen Restsatz Theorem (CRT) mit der Blakley-Multiplikation basiert, besteht zum Beispiel aus fünf hintereinander folgenden Abschnitten:

1. Erste Modulo-Operation
2. Zweite Modulo-Operation
3. Erstes modulares Potenzieren
4. Zweites modulares Potenzieren
5. Zusammenfassen der Zwischenergebnisse zum unverschlüsselten Wert

Die Berechnung der beiden modularen Potenzen stellt sich dabei als die zeitlich aufwendigste Operation dar. Ein Beispiel für die Variation des Spannungseinbruchs an einem FPGA mit diesem Algorithmus zeigt Abbildung 5.2 auf Seite 37. In dieser sind im oberen Teilbild die beiden Modulo-Operationen auf die Eingangsdaten und das anschließende binäre modulare Potenzieren gut im Profil sichtbar.

Im Vergleich dazu besteht eine Implementierung mit einem einfachen „Quadrieren und Multiplizieren“-Algorithmus und der Blakley-Multiplikation nur aus einem charakteristischen Abschnitt, weshalb sich diese Eigenschaft gut als Unterscheidung eignet. Die Leistungsaufnahme einer RSA-Verschlüsselungseinheit im Montgomery-Raum kann - abhängig von der Umsetzung - in ein bis vier Teile zerlegt werden. Je nachdem, ob der Algorithmus als Initialwert $\overline{1}$ oder $\overline{C}$ benutzt und sich die Transformation von der Multiplikation unterscheidet. Demnach liegen die meisten möglichen Abschnitte für das Erkennen eines untersuchten Algorithmus im CRT mit Montgomery-Multiplikationseinheit.

5.6 Abgleich mit einem Simulationsmodell

Mit Hilfe eines Simulationsmodells, welches die Verlustleistung für einen Algorithmus in jedem Taktschritt vorgibt, ist eine Implementierung in einer Blackbox identifizierbar. Durch die Vielfalt der Implementierungsmöglichkeiten kann kein allgemeines Verlustleistungsmodell für einen Algorithmus vorgegeben werden. Deswegen muss je nach Grad der Parallelisierung der mathematischen Operationen ein eigenes Simulationsmodell aufgestellt werden, was zu einer Datenbank von verschiedenen Implementierungsvarianten führt. Ein solches Modell macht von der Anzahl der Registeränderungen pro Schaltzeitpunkt und von „Glitches“, die durch asynchrone Logik und unterschiedliche Laufzeiten verursacht werden, Gebrauch. Mit der Voraussetzung, dass die Eingangsdaten beobachtbar und die möglichen mathematischen Operationen für den gesuchten Algorithmus in einem Modell abgebildet sind, kann so anhand von verschiedenen Modellen zwischen der Blakley- und der Montgomery-Multiplikation unterschieden werden.

Abbildung 5.4 veranschaulicht die gemittelte Messung des Spannungseinbruchs einer Blakley-Multiplikation überlagert mit einem dazugehörigen Simulationsmodell bei einer Taktfrequenz von 10 MHz. Ein Maß für die Ähnlichkeit zwischen dem Modell und der Messung kann mit dem Pearson-Korrelationskoeffizient berechnet werden. Informationen dazu finden sich in Kapitel 4.2 im Abschnitt „Correlation Power Analysis“.

5.7 Ungeeignete Angriffsvektoren

Die Detektionsmethoden, die im vorherigen Unterkapitel zur Diskussion standen, zeichnen sich durch ihren generischen Ansatz aus. Eine Ausnahme stellt dabei jedoch der Einsatz von Simulationsmodellen dar. Generisch bedeutet in diesem Kontext, dass die Vorgangsweise der Detektionsmethoden wenig von der Implementierung abhängig ist. Ohne speziellen Gegenmaßnahmen, wie zum Beispiel das Randomisieren des Verschlüsselungsvorgangs, treten für Extremwerte immer dieselben detektierbaren Operanden für dieselben Operationen auf, auch wenn der Rechenvorgang in einem FPGA in einer anderen Anzahl von Taktzyklen unterteilt ist. Diese Eigenschaft trifft genauso für die Abschnittsanalyse, die Rechenzeitanalysen und die Variation der Eingangsdaten zu.

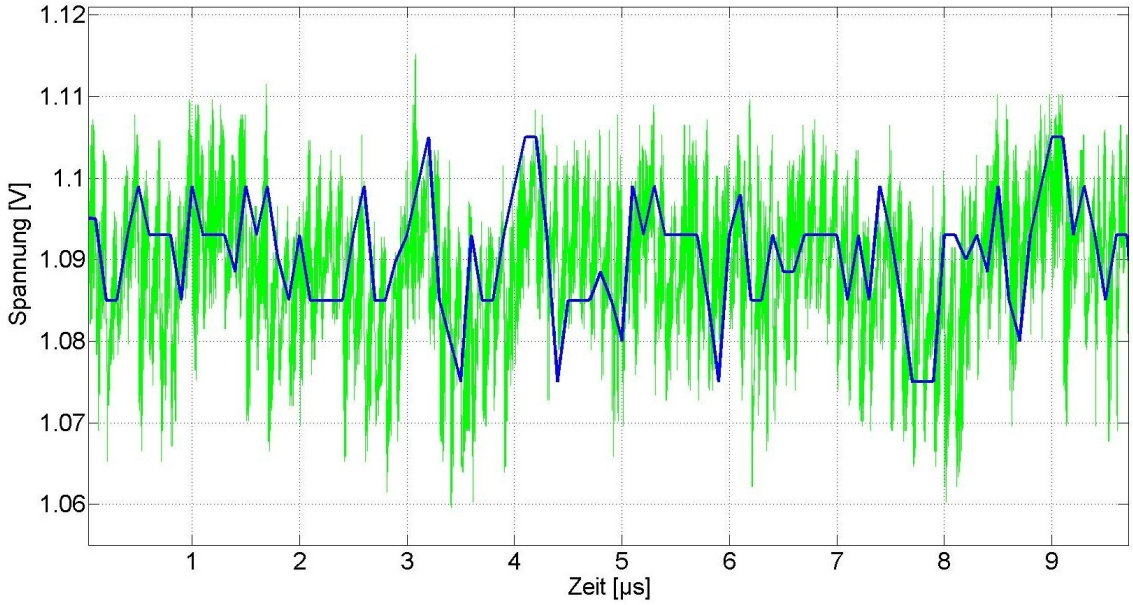

Abbildung 5.4: Überlagerung des realen Spannungseinbruchs durch die Blakley-Multiplikation in Grün mit dem in Blau eingezeichneten, generierten Simulationsmodell

Um unbekannte Algorithmen zu detektieren, eignen sich nicht alle Seitenkanalangriffsmethoden: Template-Attacken benötigen aus einem baugleichen, mit demselben Programmierfile konfigurierten Device generierte Templates, die dem generischen Detektionsansatz widersprechen. Ein Angriff über „High-Order Differential Power Analysis“ setzt ebenfalls Detailinformationen voraus, die für eine Detektion meist nicht zur Verfügung stehen.

Abbildung 5.5 verschafft einen Überblick über die verschiedenen Detektionsmethoden in der Form eines Blockdiagramms. Einerseits ordnet das Bild die Detektionsvorgänge der jeweiligen Kategorie zu und andererseits verknüpft es die daraus folgende Interpretation bei positiver Identifikation.

5.8 Analyse der Schutzmaßnahmen gegen Detektionsmethoden

Dieser Abschnitt hilft den Einsatz von verschiedenen Maßnahmen gegen die vorgestellten Detektionsmethoden aus den Unterkapiteln 5.2, 5.3, 5.4, 5.5 in ihrer Anwendung zu verstehen. Dabei erarbeitet dieses Kapitel Sicherheitsmechanismen und Verschlüsselungsimplementierungen, um sich vor einer Detektion zu schützen. Der Nachteil, dass ein seitenkanalresistenter VHDL-Code zu zusätzlicher Logik führt, gilt im Folgenden für alle vorgestellten Methoden.

5.8.1 Verlängerung der Ausführungszeit mit zusätzlichen Operationen

Durch die Kenntnis der Ausführungszeit einer Operation im „Quadrieren und Multiplizieren“-Algorithmus, unabhängig davon, ob es sich um eine Montgomery- oder eine Blakley-Multiplikation handelt, besteht die Möglichkeit, das Hamming-Gewicht des Schlüssels und in weiterer Folge die Bitlänge der Verschlüsselungseinheit zu erkennen. Außerdem lässt die zeitliche Variation bei der Veränderung der Eingangsdaten auf einen CRT-RSA-Algorithmus rückschließen.

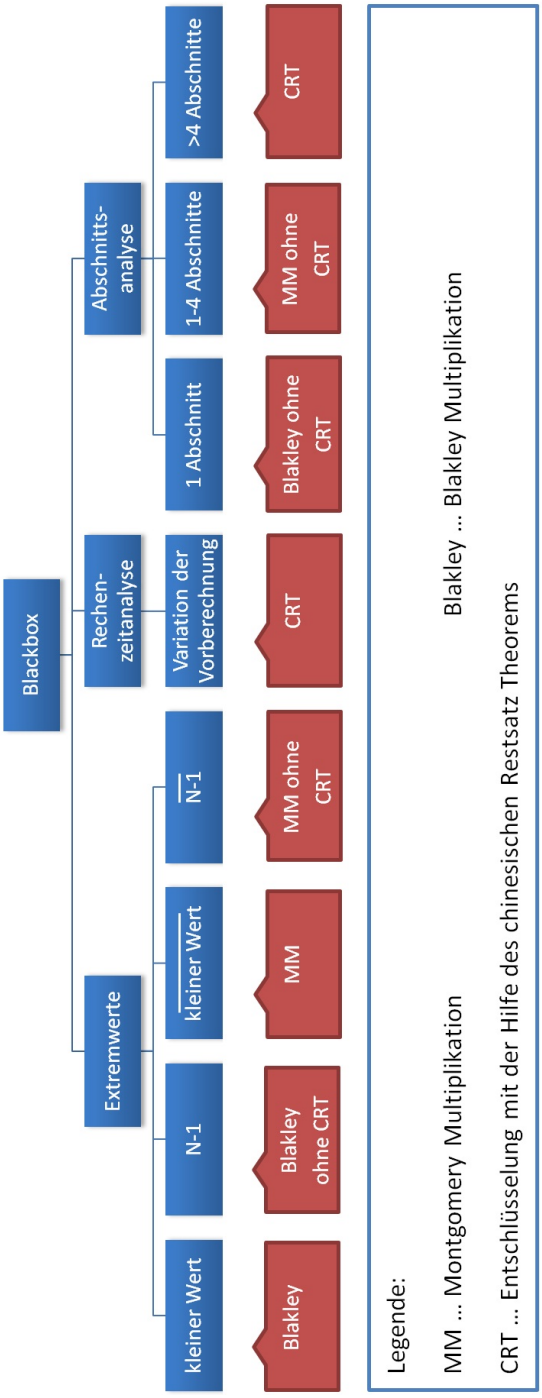

Abbildung 5.5: Überblick der Detektionsmethoden

Um diese zeitliche Beobachtung zu verhindern, hilft es, zusätzliche Operationen in den Code einzubauen, welche zu einem konstanten zeitlichen Verhalten führen. Diese hinzugefügten Operationen dürfen sich nicht stark von den anderen unterscheiden, um keine neuen, womöglich stärkeren Seitenkanäle zu erschaffen. Neben der zusätzlichen Logik, die im Zuge dieser Maßnahme erforderlich ist, führt die künstliche Verlängerung der Ausführungszeit zu dem längst möglichen zeitlichen Ausführungsfall und dadurch zu einer niedrigeren Rechenzeitperformance.

Eine dieser Gegenmaßnahmen ist unter dem englischen Schlagwort „Multiply Always“ [WWM11] bekannt und kann der Methode des Versteckens zugeordnet werden. Abhängig vom Schlüssel wird in einer ungeschützten Implementierung entweder nur quadriert oder zusätzlich auch multipliziert. Findet der Angreifer einen Unterschied in der Ausführung dieser beiden Operationen, so kann er den Schlüssel einfach rekonstruieren. Ein eingesetzter Verschlüsselungsvorgang, der unabhängig vom Schlüssel jede Runde multipliziert, kann diese Beobachtung verhindern. Eine mögliche Berechnungsvorschrift dazu nennt sich „Montgomery Power Ladder“ (Algorithmus 4.1). Dieser Mechanismus unterbindet keinen Angriff über Extremwerte, da weiterhin das jeweilige Ergebnis aus der vorherigen Runde den nächsten Operanden vorgibt, was im Zuge seiner Ausführung auf die vorherige Operation rückschließen lässt.

5.8.2 Randomisieren von Verschlüsselungsparametern

Wie im vorherigen Teil gezeigt, nimmt durch die Vorgabe von gewissen Extremwerten - abhängig vom Verschlüsselungsalgorithmus - die momentane Verlustleistung des untersuchten FPGAs markante Werte an. Dies kann für die Unterscheidung von Operationen, wie dem Quadrieren oder dem Multiplizieren, hilfreich sein. Wenn eine Manipulation der Eingangsdaten in der Verschlüsselungsmethode unterbunden werden kann, so ist ein solcher Angriff durch reines Beobachten beinahe auszuschließen.[1] Es gibt verschiedene Gegenmaßnahmen, um einen solchen Angriff über Extremwerte zu verhindern:

Eine Maßnahme erfolgt über die Einschränkung des möglichen Zahlenraums auf einen Bereich ohne Extremwerte. Diese Verminderung der verwendbaren Verschlüsselungswerte minimiert die Symbole für die Übertragung nur geringfügig.

Durch das Maskieren der Nachricht, auch unter dem Begriff „Message Blinding“ bekannt, wird der Eingangswert mit der Zahl m_1, die aus einem zufällig generierten Wert r, der teilerfremd zu N ist, und dem öffentlich bekannten Schlüssel (e, N) besteht, multipliziert und anschließend nach dem Verschlüsselungsvorgang wieder auf den richtigen Wert über die inverse Zahl m_2 korrigiert. Diese zwei Zahlen werden wie folgt berechnet [BK08]:

$$m_1 = r^e \mod N \tag{5.1}$$

$$m_2 = r^{-1} \mod N \tag{5.2}$$

Das strikte Einsetzen in den Entschlüsselungsalgorithmus zeigt die Richtigkeit dieser Faktoren.

$$M = (m_1 \cdot C)^d \cdot m_2 \mod N = r^{\overbrace{e \cdot d}^{1}} \cdot C^d \cdot r^{-1} \mod N = C^d \mod N \tag{5.3}$$

Durch diesen Zwischenschritt wird der Rechenvorgang so manipuliert, dass durch die Vorgabe der Eingangsdaten des Angreifers keine Rückschlüsse aus der Messung gezogen werden können. Genauer gesagt, wird ein Extremwert vor der Entschlüsselung auf eine andere Zahl transformiert, die eine Detektion des Algorithmus dadurch verhindert.

Analog dazu bietet die Arbeit [Bau12] die Möglichkeit - neben dem Verstecken der Nachricht im RSA-Algorithmus - auch den Exponenten, um einen bestimmten Faktor zu verschieben, ohne eine Auswirkung auf das Ergebnis zu verursachen. Bei diesem Faktor handelt es sich um die eulersche

[1] Es ist nur sehr unwahrscheinlich, dass man so einen Extremwert, der sich als verschlüsselte Nachricht ausgibt, zufällig beobachten kann.

Funktion $\varphi(N)$, die für die Generierung des Schlüssels verantwortlich ist und sich aus den beiden Primzahlen ergibt ($\varphi(N) = (p-1) \cdot (q-1)$). Die Rückwandlung aus dem verschlüsselten Zustand eines Werts kann immer dann vollzogen werden, wenn der geheime Schlüssel die Bedingung $1 \equiv e \cdot d \mod \varphi(N)$ erfüllt. Solange sich der geheime Schlüssel d nur um ein Vielfaches der eulerschen Funktion additiv ändert, bleibt diese Vorgabe erfüllt. Der neue Schlüssel $\overline{d}$ berechnet sich also mit einer ganzzahligen positiven Zufallszahl r zu

$$\overline{d} = d + r \cdot \varphi(N). \tag{5.4}$$

Diese Gegenmaßnahme erschwert eine statistische Messung und kann die Schlüssellänge für jeden Verschlüsselungsdurchgang ändern. Der Mechanismus schützt jedoch nicht vor einem Detektionsangriff mit speziellen Nachrichten, da die Operanden weiterhin Informationen preisgeben können.

Das Randomisieren des Exponenten und der Nachricht wird in der Literatur auch „Maskierung“ [Zef07] bezeichnet.

5.8.3 Maßnahmen gegen statistische Auswertungen

Ein geringes Signal-Rausch-Verhältnis der Messsignale erhöht den Aufwand eines Seitenkanalangriffs. Das gilt vor allem für die Abschnittsanalyse, die versucht, Muster in den unterschiedlichen Rechenvorgängen zu finden, und die Methode der Variation der Eingangsdaten.

Um mehrere Messungen zu mitteln und anschließend auszuwerten, benötigt man Messvorgänge, die zeitlich exakt übereinstimmen. Durch starken Jitter in der Frequenz und zufällige Pausen während des Entschlüsselungsalgorithmus kann die Implementierung eine Mittelung der Messung stören. Zufällig eingefügte Taktzyklen, die den Verschlüsselungsvorgang unterbrechen, sollten im Vergleich zu den anderen Berechnungen nicht unterscheidbar sein. Die Initialisierung eines Pseudozufallsgenerators, der für solche randomisierte Abläufe benötigt wird, sollte keinesfalls auf der Basis von Sicherheit durch Obskurität („Security by Obscurity“) stehen, sondern beispielsweise auf interne Zustände, wie den geheimen Schlüssel und auf einem beliebigen, nicht vorhersehbaren Wert beruhen.

Eine hohe Taktfrequenz erschwert eine genaue Analyse der Verlustleistung eines FPGAs, da sich Störungen, wie unerwünschte harmonische Schwingungen, stärker auf das sonst stabile Messsignal auswirken.

Besonders Schutzmaßnahmen, die versuchen durch bestimmte Tätigkeiten die Verbindung zwischen den gemessenen Seitenkanal und der tatsächlich ausgeführten Aktivität zu minimieren, gelten für einen Angriff mit Extremwerten als eher ineffektiv. Die Schwierigkeit besteht vor allem darin, die deutlichen Unterschiede, die durch die Vorgaben von speziellen verschlüsselten Nachrichten auftreten, zu glätten. Durch zusätzliches Rauschen in Form von gleichzeitig aktiver Logik und die Unterteilung der Ausführungsschritte in zusätzliche Taktzyklen kann trotzdem versucht werden, durch reine Veränderung im VHDL-Code, diese Schutztechnik umzusetzen.

Alle aufgezählten Vorgangsweisen erschweren in erster Linie einen erfolgreichen Angriff, können diesen aber nicht vollkommen unterbinden und ausschließen.

5.8.4 Limitierung von Sicherheitsmechanismen

Anders als in der Kryptographie, in der die Verschlüsselung als sicher gilt, solange kein effektiver mathematischer Angriffsvektor existiert, ist die Sicherheit gegen eine Seitenkanalanalyse besonders von den Fähigkeiten des Angreifers und der Messumgebung abhängig. Viele Faktoren, wie die Güte des Seitenkanals, die Verarbeitung der Messwerte oder das zugrundeliegende Modell entscheiden oft über den Erfolg oder Misserfolg eines Angriffs. Aus diesem Grund lässt sich keine universelle Gegenmaßnahme definieren.

Schlecht implementierte Gegenmaßnahmen, wie Zwischenoperationen die nur zeitliche Löcher füllen, aber einfach detektierbar sind, können zu zusätzlichen und womöglich stärkeren Seitenkanalinformationen führen und sollten deswegen vermieden werden.

Laut [Rüd09, S. 54-56] stehen viele Geräte, in denen sich kryptographische Algorithmen finden, durch die Massenproduktion unter starkem Kostendruck und sind daher in ihren Ressourcen begrenzt. Das führt dazu, dass es oft nicht möglich ist, ausreichende Schutzfunktionen umzusetzen. Ein weiteres Problem liegt vor allem bei Produkten mit langer Lebenszeit vor. Auch wenn heute kein Angriff auf eine bestimmte - nach dem Stand der Technik sichere - Gegenmaßnahme bekannt ist, kann sich dies über die Zeit durch verbesserte Methoden und neue Angriffsvektoren ändern.

6 Seitenkanalangriffe auf ausgewählte Implementierungen

Dieses Kapitel diskutiert verschiedene Seitenkanalangriffe auf ausgewählten RSA-Implementierungen, die auf einem „Atlys“-Entwicklungsboard abgearbeitet werden. Neben „Simple Power Analysis“ (SPA) und „Differential Power Analysis“ (DPA) steht ebenfalls die Rechenzeitanalyse im Fokus.

Wie bereits in Kapitel 5.4 vorgestellt, nutzt der Angriff über SPA Extremwerte für die Detektion. Aus diesem Seitenkanal lassen sich neben dem Algorithmus auch Schlüsselinformationen extrahieren. Auf die Interpretation dieser Messergebnisse wird deswegen in vier Algorithmen eingegangen und anhand von verschiedenen praktischen Implementierungen verifiziert. Der Seitenkanalangriffsvektor DPA verwendet die Korrelationsmethode, die die Messung mit einem erstellten Leistungssimulationsmodell abgleicht und der richtigen Operation zuordnet. Die Extraktion des Schlüssels kann auch in Implementierungen mit dem chinesischen Restsatz Theorem über die Rechenzeitanalyse erfolgen und wird deswegen in dem folgenden Kapitel eingesetzt.

Ein modularer Aufbau in VHDL mit verschiedenen Komponenten unterstützt das Einbinden von unterschiedlichen Algorithmen. Außerdem kommt immer dieselbe Kommunikationseinheit für die Ein- und Ausgabe von Daten zum Einsatz. Abbildung 6.1 zeigt ein Beispiel dieser Grundstruktur, wobei die Module mit „Load“- und „Busy“-Signalen synchronisiert sind und der Trigger für das Oszilloskop mit der „Busy“-Leitung der Verschlüsselungseinheit verbunden ist.

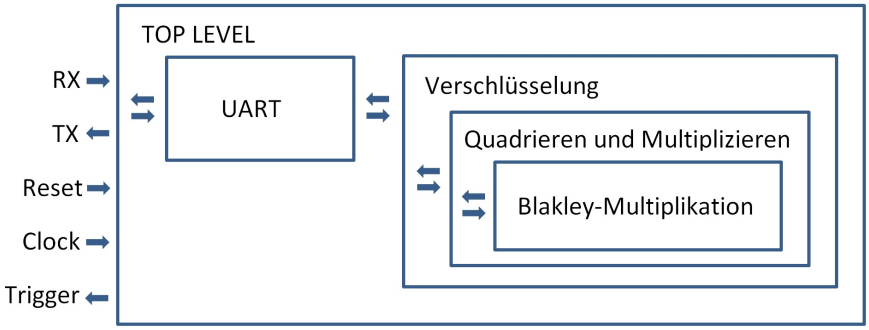

Abbildung 6.1: Struktureller Aufbau in VHDL mit der Einbindung der Blakley-Multiplikation im „Quadrieren und Multiplizieren“-Algorithmus

6.1 „Quadrieren und Multiplizieren“-„Blakley“-Algorithmus

Der RSA-Verschlüsselungsalgorithmus, welcher als erster auf Seitenkanäle analysiert wurde, verwendet zum modularen Potenzieren der Daten einen 128 Bit „Quadrieren und Multiplizieren“-Algorithmus (Kapitel 3.2.2). Als Basisrecheneinheit benutzt diese Rechenvorschrift die Blakley-Multiplikation (Kapitel 3.4). Der grobe strukturelle Zusammenhang kann Abbildung 6.1 entnommen werden.

In diesem Fall führt eine Rechenzeitanalyse nicht zum Ziel, da die Ausführungszeit unabhängig von den Eingangsdaten ist und nur vom verwendeten Schlüssel abhängt. Mit „Simple Power Analysis“ und „Differential Power Analysis“ existieren hingegen gute Möglichkeiten, Informationen über den Schlüssel zu sammeln.

6.1.1 Rechenzeitanalyse

Wie bereits erwähnt, kann aus der Rechenzeit des Algorithmus nicht direkt auf den verwendeten Schlüssel Rückschluss gezogen werden. Es besteht jedoch eine Proportionalität für jedes gesetzte Bit des geheimen Schlüssels und der zusätzlichen Ausführungszeit der Verschlüsselungsmethode. Gesetzte Bits im Schlüssel verursachen neben der notwendigen Quadrier-Operation eine weitere Multiplikation. Diese Eigenschaft führt zu einer variablen Rechenzeit, die das Hamming-Gewicht des Schlüssels verrät. Wenn für eine n Bit Implementierung m berechnete Operationen gemessen werden, so ergibt sich aus der Differenz $m-n$ die Anzahl der gesetzten Bitstellen im Schlüssel. Aus der Kenntnis dieses Gewichts kann der Schlüsselraum eingeschränkt und ein möglicher detektierter Schlüssel überprüft und korrigiert werden. Anstatt der 2^n verschiedenen Schlüsselmöglichkeiten existiert nur noch der kleinere Schlüsselraum der Größe

$$\binom{n}{(m-n)} \text{ beziehungsweise } \frac{n!}{(m-n)! \cdot (2 \cdot n - m)!} \,. \tag{6.1}$$

6.1.2 Simple Power Analysis

Durch das Beobachten des Spannungseinbruchs an der FPGA-Versorgung und der Einspeisung von Extremwerten anstelle einer korrekt verschlüsselten Nachricht ist es einem Angreifer möglich, in nur einer Messaufnahme den geheimen Schlüssel zu rekonstruieren. Um diesen Extremwert messen zu können, muss es dem Analysten möglich sein, die chiffrierte Nachricht für die Entschlüsselung vorzugeben oder über „Fault Injection“ zu infiltrieren.

Besonders zwei Extremwerttypen zeigen ein deutliches Profil in der Verlustleistung:

- Ein besonders kleiner Wert als verschlüsselter Text (jedoch größer als eins)
- Die größtmögliche Zahl beziehungsweise der um eins minimierte Zahlenwert des öffentlichen Modulus

Die richtige Interpretation der sichtbaren Ergebnisse im Spannungsprofil hängt von der verwendeten Extremwertmethode ab. Ein besonders leicht erkennbares Seitenkanalsignal ergibt sich, wenn kleine Zahlen als Eingangswerte für die Entschlüsselung vorgegeben werden. Die Multiplikationsmethode des Blakley-Verfahrens besitzt dadurch einen ebenfalls kleinen Operanden, der zu einer

deutlich minimierten momentanen Verlustleistung führt. Folgt dem Quadrieren eine Multiplikation, so weist dies auf eine binäre Eins im Exponenten hin, was gleichbedeutend mit einer binären Eins an dieser Schlüsselposition ist. Mit dieser Methode kann so der Schlüssel aus der Messung rekonstruiert werden.

Betrachtet man die ersten Bits im Exponenten für diesen Extremwert in der Ausführung, sieht man keine deutlichen Unterscheidungsmerkmale zwischen Multiplizieren und Quadrieren. Das hat den Hintergrund, dass sich zuerst ein Zwischenergebnis mit einem gewissen Gewicht bilden muss, welches dann die Unterscheidung erst möglich macht. In Kombination mit einer „Bruteforce-Attacke“ lassen sich diese nicht detektierbaren Bits erahnen.

Natürlich kann ein Zwischenergebnis nach dem Multiplizieren oder Quadrieren auch in einem ähnlich kleinen Umfeld liegen, was jede folgende Operation wie eine Multiplikation aussehen lässt. Um diesen eher seltenen Fall zu verhindern und zu erkennen, kann ein anderer kleiner Eingangswert, der verschiedenartige Zwischenergebnisse produziert, helfen. Abbildung 6.2 macht diese messbaren Seitenkanalinformationen sichtbar und beschriftet sie mit den binären Werten des geheimen Schlüssels. Die Buchstaben Q und M stehen für Quadrieren beziehungsweise Multiplizieren und benennen die jeweilige Operation in der Grafik.

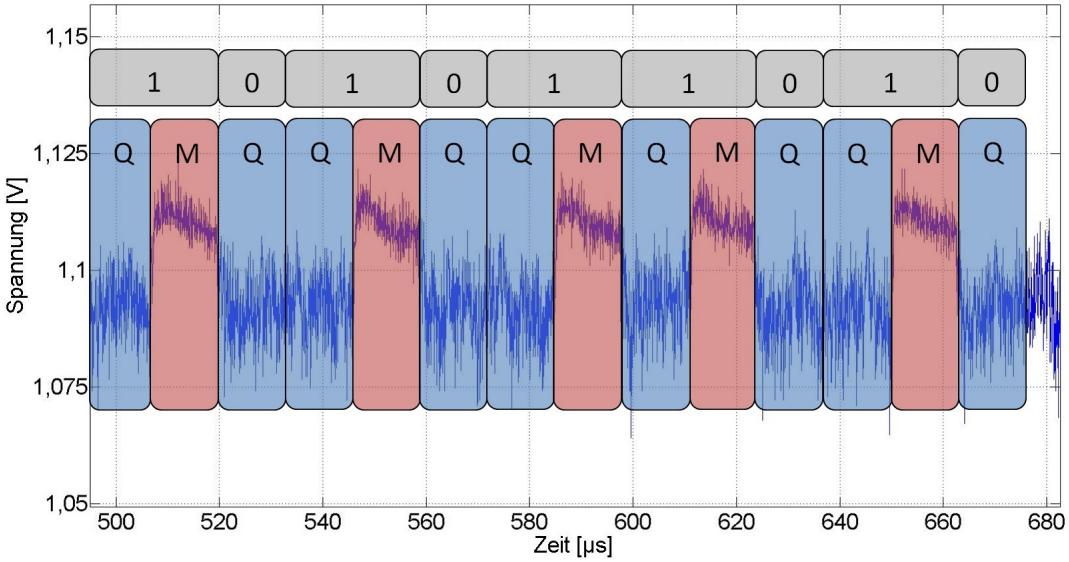

Abbildung 6.2: Extremwertanalyse - mit kleinen Werten

Die andere Extremwertmethode, die mit dem größten erlaubten Wert arbeitet, nutzt einen mathematischen Sonderfall aus. Für eine ganze Zahl N größer eins gilt immer

$$(N-1)\cdot(N-1) \mod N = 1. \tag{6.2}$$

Diese Variable N repräsentiert in RSA den öffentlichen Modulus und ist somit bekannt. Wählt ein Angreifer nun als verschlüsselte Nachricht $N-1$ kann nur durch eine einzige Messaufnahme der Schlüssel aus dem Spannungsprofil abgelesen werden. Ein kurzes Beispiel für einen privaten Schlüssel ($b = [b_1, b_0] = 10_2$) soll diesen Fall verdeutlichen:

Im ersten Schritt betrachtet man das höchstwertigste Bit im geheimen Schlüssel ($b_1 = 1$). Laut der „Quadrieren und Multiplizieren“-Vorschrift bedeutet dies, dass zuerst der Wert 1 mit sich selbst quadriert und anschließend mit $N-1$ multipliziert wird. Das nächstfolgende Schlüsselbit

b_0 ist nicht gesetzt und verlangt deshalb nur das Quadrieren des Zwischenwerts. Dieser Vorgang minimiert das Ergebnis abermals auf 1 und bringt es auf den Ausgangspunkt zurück.

Das von der letzten Schlüsselstelle abhängige Endergebnis kann zwei verschiedene Werte annehmen:

$$(N-1)^b \mod N = \begin{cases} 1, & \text{falls } b_0 = 0 \\ N-1, & \text{sonst} \end{cases} \tag{6.3}$$

Mit dieser Erkenntnis lässt sich das niederwertigste Schlüsselbit alleine aus dem Endergebnis herauslesen.

Diese Berechnungen, verursacht durch den Extremwert, spiegeln sich im Spannungsprofil wider. Drei Fälle können auf diese Weise unterschieden werden:

A) Quadrieren des binären Werts 1 - Kaum sichtbarer Spannungseinbruch

B) Multiplizieren des binären Werts 1 mit $N-1$ - Geringfügig sichtbarer Spannungseinbruch

C) Quadrieren des Werts $N-1$ - Stark sichtbarer Spannungseinbruch

Mit diesem Extremwert und SPA erkennt man eine Multiplikation in der darauffolgenden Operation (Quadrieren des Werts $N-1$), die einen deutlich sichtbaren Signalausschlag in der Messung hervorruft. Der Vorteil gegenüber der Extremwertmethode mit kleinen Werten besteht darin, dass alle Bits des Schlüssels sofort erkannt werden können. Die Grafik 6.3 zeigt dieses Verhalten und verdeutlicht, welcher der drei Fälle in welchem Teilabschnitt erkennbar ist. Außerdem ordnet dieses Bild die erkannten Operationen dem verwendeten binären privaten Schlüssel zu.

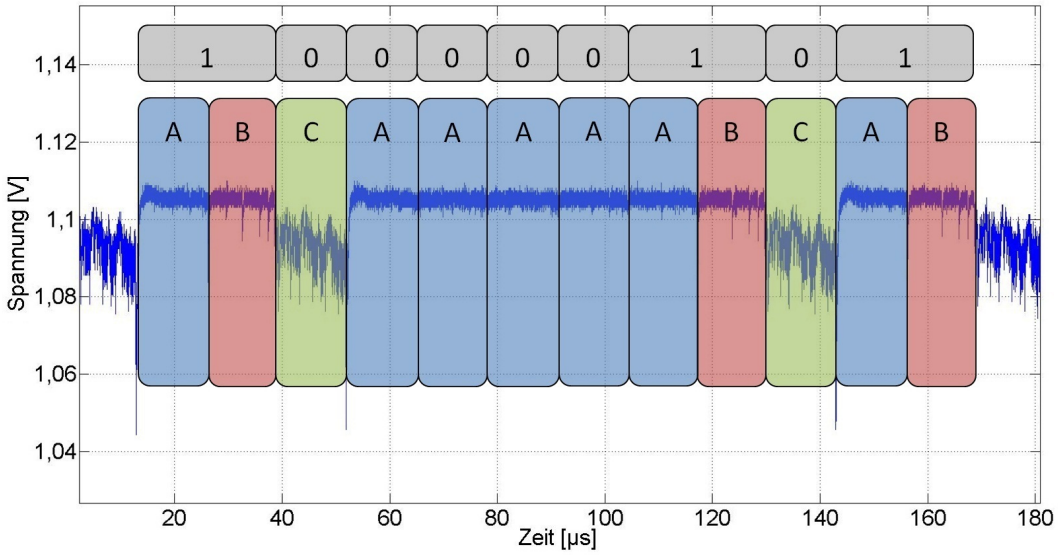

Abbildung 6.3: Extremwertanalyse - mit größter möglicher Zahl

Da nach einer Multiplikation keine weitere Multiplikation folgt, treten diese drei Fälle nur in einer bestimmten Reihenfolge auf. Dieses Verhalten kann dazu verwendet werden, die beiden ähnlichen Abschnitte A und C zu unterscheiden. Vor Fall B wird entweder Fall A oder Fall C ausgeführt. Nach Fall B folgt immer Fall C.

6.1.3 Differential Power Analysis

Die bisher vorgestellten Angriffe basieren eher auf dem „Quadrieren und Multiplizieren“-Algorithmus und greifen nur indirekt durch das Erkennen von kleinen und großen Operanden auf die Eigenschaften der Blakley-Multiplikation zu. Der in diesem Abschnitt beschriebene Angriff arbeitet mit statistischen Methoden, um detaillierter in die Blakley-Rechenvorschrift einzugreifen.

Um zu zeigen, dass sich die Messaufnahmen mit denselben Eingangsdaten ähneln, wurde die Differenz von zwei Aufnahmen mit gleichem beziehungsweise unterschiedlichem Schlüssel miteinander verglichen. Es stellte sich heraus, dass die durch dieselbe Berechnung entstehenden Spannungseinbrüche große Ähnlichkeiten aufweisen. Dieser Erstversuch wurde bereits in Kapitel 4.2 und in der Abbildung 4.3 in Form einer Templateattacke festgehalten.

Mit der Gewissheit, dass diese Spannungsschwankungen einer Systematik folgen, wurde ein C-Programm entwickelt, welches jeden Rechenschritt im Blakley-Algorithmus bitweise simuliert. Die Blakley-Operation teilt sich für die untersuchte 128 Bit Implementierung in 128 Taktschritte. Daher muss ein mögliches Modell für jeden Rechenvorgang der Basiseinheit genauso viele geschätzte Spannungseinbrüche vorgeben. Das in dieser Arbeit generierte Simulationsmodell wurde auf Basis des VHDL-Codes 6.1 entwickelt und stellt eine Blakley-Multiplikation dar.

VHDL-Code 6.1: Blakley-Multiplikation

```
1  p_modmult: process(s_clk)  --Blakley
2  Methode variable s_between : UNSIGNED ((3*bitwidth -3) downto 0);
3  begin
4  if rising_edge(s_clk) then
5    if s_rst='1' then
6      s_busy<='0';
7    else
8      if s_busy='0' and s_ld='1' then    --Laden
9        s_busy<='1';
10       s_between:=(OTHERS=>'0');
11       act_count<= 0;
12       s_state1<=state1_i;
13       s_state2<=UNSIGNED(state2_i);
14       s_modul<=UNSIGNED(modul_i);
15     elsif s_busy='1' then
16       s_between:=s_between sll 1;       --Shift y=(y*2)
17       if s_state1(bitwidth-1-act_count)='1' then --Addition y=y+a_{k-1-i}*b
18         s_between:=s_between+s_state2;
19       end if;
20       if s_between>=s_modul then
21         s_between:=s_between-s_modul;  --Subtraktion 1
22       end if;
23       if s_between>=s_modul then
24         s_between:=s_between-s_modul;  --Subtraktion 2
25       end if;
26       if act_count=bitwidth-1 then
27         s_busy<='0';                   --Ende
28         s_state_o<=std_logic_vector(s_between((bitwidth-1) downto 0));
29       else
30         act_count<=act_count+1;        --Naechste Runde
31       end if;
32     end if;
33   end if;
34 end if;
```

Im Zuge der Simulation der Blakley-Implementierung wurde versucht, mehrere algorithmische Verhalten mit dem Spannungseinbruch abzugleichen. Einen Überblick dieser getesteten und simulierten Verhalten gibt folgende Aufzählung:

- Bitwechsel vor und nach einem Taktzyklus von *s_between*
- Bitwechsel zwischen jeder Operation auf *s_between* innerhalb eines Takts
- Wie viele if-Statements im Vergleich zum vorherigen Takt ihren logischen Zustand ändern

Die beste Übereinstimmung zeigt die Kombination aus den Änderungen der Gültigkeit von if-Statements und einem zusätzlichen Faktor, der dann ins Gewicht fällt, wenn sich viele Bits zwischen den Rechenoperationen gleichzeitig ändern. Im Fall der Blakley-Multiplikation unterscheidet man nach dem VHDL-Code fünf Fälle[1]:

1. Shift
2. Shift, Addition
3. Shift, Addition, Subtraktion 1
4. Shift, Addition, Subtraktion 1, Subtraktion 2
5. Shift, Subtraktion 1

Aus diesen Ablaufszenarien ergeben sich 25 mögliche Änderungen, die jeweils nach ihrer Auswirkung im Modell gewichtet sind. Zu diesem Faktor wird anschließend der weitere Faktor für veränderte Bits innerhalb der Operationen aufgeschlagen.

Die Analyse wurde so gestaltet, dass über das Programm MATLAB® das Oszilloskop angesteuert und der Verschlüsselungsalgorithmus im FPGA gestartet werden konnte. Damit kann ein generierter Mittelwert aus den aufgenommenen Spannungseinbrüchen am FPGA in mehreren Messreihen automatisiert erzeugt werden, der das Rauschen durch Mittelung reduziert und dadurch das SNR erhöht. Abbildung 5.4 auf Seite 41 zeigt in Grün eine über 200 Samples gemittelte Messung, sowie in Blau das durch die Simulation generierte zugehörige Modell. Vergleicht man in dem Bild den Verlauf der Messdaten mit den Modelldaten, erkennt man eine grobe Übereinstimmung in weiten Bereichen.

Für die Bewertung der Ähnlichkeit der beiden Kurven hilft die Kreuzkorrelation. Eine kurze positive Signalspitze im übereinstimmenden Korrelationszeitpunkt signalisiert, dass sich beide Signale gut decken. Im rechten Teil der Abbildung 6.4 erkennt man diese Spitze für die Messung und das Modell aus Abbildung 5.4. Im Vergleich dazu veranschaulicht das linke Teilbild die Kreuzkorrelation für ein falsch angenommenes Modell. In diesem Fall kann deutlich das Fehlen des Korrelationsmerkmals in der Mitte beobachtet werden. Neben der Zuordnung der Signale zu der zugehörigen Operation, kann die Korrelationsspitze für die Synchronisation mit dem nächsten Berechnungsschritt eingesetzt werden.

Diese Angriffsmethode erfordert ein iteratives Vorgehen, da das Ergebnis aus der vorherigen für die darauffolgende Operation im Simulationsmodell benötigt wird. Abhängig vom geheimen

[1] Bemerkung: Der Fall „Shift, Subtraktion 1, Subtraktion 2“ kann nicht eintreten, da die Shift-Operation die Zahl in diesem Ausmaß nicht ändern kann.

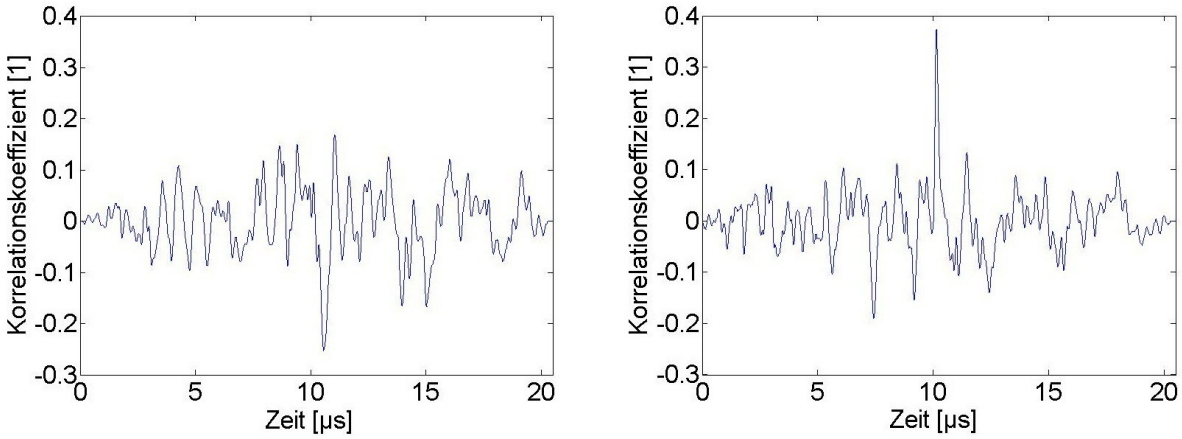

Abbildung 6.4: Kreuzkorrelation (Blakley) - Links: falsches Modell - Rechts: richtiges Modell

Schlüssel bedeutet dies für eine 128 Bit Verschlüsselung zwischen 128 und 256 iterative Schritte in der Simulation. Folgefehler im iterativen Vorgang weisen auf die Wahl eines falschen Modells in einem vorangegangenen Schritt hin. Fehler in der Interpretation liegen dann vor, wenn sich die jeweils zwei möglichen Modelle (Quadrieren und Multiplizieren) nur wenig voneinander unterscheiden. Abbildung 6.5 zeigt eine falsche Zuordnung der ausgeführten Operation zum simulierten Modell und einen darauffolgenden erkannten Fehler, der zu einer Korrektur führt. Da verschiedene Verschlüsselungseingangsdaten zu unterschiedlichen Simulationsergebnissen und Spannungseinbrüchen führen, kann auch auf diesem Wege die falsche Zuordnung der Messung zum Modell detektiert und umgangen werden.

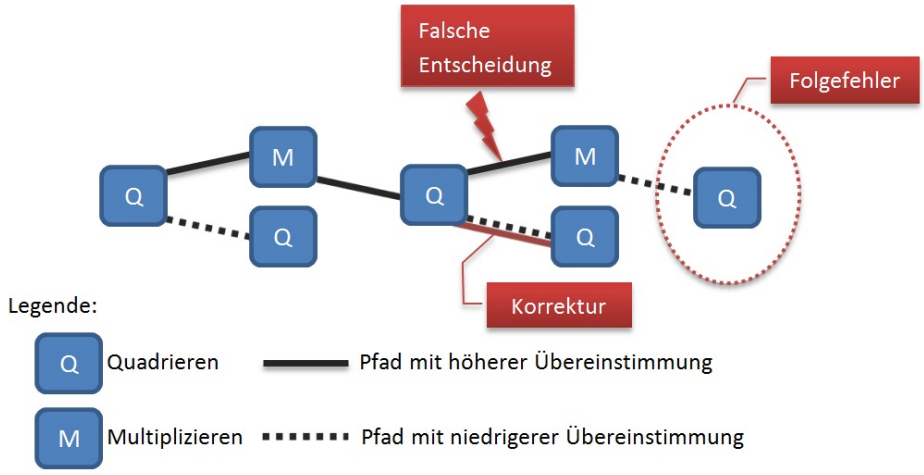

Abbildung 6.5: Fehlerkorrektur

Ein großer Vorteil gegenüber der bereits vorgestellten „Simple Power Analysis“, die auf Extremwerte beruht, ist, dass hier nur der am Eingang vorliegende verschlüsselte Wert für das Modell bekannt und nicht modifizierbar sein muss. Mit dem Erkennen von Folgefehlern lassen sich schnell Falschannahmen erkennen und korrigieren. Der Nachteil beim Einsatz der „Differential Power Analysis“ liegt darin, dass der Beobachter für einen erfolgreichen Angriff die Kenntnis über

den VHDL-Code oder zumindest über ein Modell für die typische Abbildung des verwendeten Algorithmus in FPGA-Strukturelemente benötigt und viele Verschlüsselungsvorgänge mitverfolgen muss. Verfügt der Angreifer über eine Sammlung von Simulationsmodellen mit mehreren möglichen FPGA Implementierungen des untersuchten Algorithmus, kann auch ohne VHDL-Implementierungsdetails ein Vergleich mit der tatsächlichen Messung erfolgreich sein.

Die statistische Aufnahme über mehrere Messvorgänge verursacht bei einem geringen SNR eine sehr zeitintensive Analyse. Da die Blakley-Multiplikationseinheit auch für das Quadrieren eingesetzt wird, unterscheidet sie sich in der Ausführung nur von den geladenen Operanden. Deshalb muss die chiffrierte Nachricht in dem gewählten Simulationsmodell bekannt sein.

6.2 „Chinesischer Restsatz“-„Blakley“-Algorithmus

Der zweite Algorithmus, der in dieser Arbeitet betrachtet wurde, basiert auf dem chinesischen Restsatz Theorem. Das Verfahren wird mit CRT-RSA abgekürzt und beschleunigt den Verschlüsselungsvorgang im Vergleich zum vorherigen Verfahren. Dies wird durch die Weiterverwendung der beiden Primzahlen p und q, die für die Generierung des Schlüssels erforderlich sind, möglich. Genauere Informationen dazu befinden sich im Abschnitt 3.2.1. Als Basisalgorithmus für den benötigten Exponenzierer steht der „Quadrieren und Multiplizieren“-„Blakley“-Algorithmus aus Kapitel 6.1 zur Verfügung.

Um die folgenden verschiedenen Seitenkanalangriffe zu verstehen, soll nochmals kurz der Ablauf der Entschlüsselung mit CRT-RSA und die dafür benötigten Berechnungen betrachtet werden:

1. Generierung des Primzahlpaars durch einen Zufallsgenerator
2. Erzeugung des öffentlichen und privaten Schlüssels ((e, N) bzw. d) (siehe Schlüsselgenerierung im Kapitel 3.2)
3. Berechnung der modularen Inversen $q_{inv} = \frac{1}{q} \mod p$
4. Kalkulation der beiden Subschlüssel durch $d_p = d \mod (p-1)$ und $d_q = d \mod (q-1)$
5. Reduzieren des Eingangswerts C, welcher die verschlüsselte Nachricht repräsentiert, auf die Bitbreite des modularen Exponenzierer $C_p = C \mod p$ und $C_q = C \mod q$
6. Berechnen der beiden Ausdrücke $M_p = C_p^{d_p} \mod p$ und $M_q = C_q^{d_q} \mod q$
7. Ermitteln des Faktors $h = q_{inv} \cdot (M_p - M_q) \mod p$
8. Bestimmen des unverschlüsselten Texts $m = M_q + h \cdot q$

Man unterscheidet dabei zwischen Berechnungen, die bereits im Vorhinein durchführbar sind und welche, die die verschlüsselte Nachricht als Abhängigkeit vorweisen und somit erst zur angeforderten Zeit ausgeführt werden. Die Schritte Eins bis Vier zählen dabei zu den vorausberechenbaren Teilen und sind daher selten beobachtbar. Hingegen sind die Berechnungen Fünf bis Acht für jeden Verschlüsselungsvorgang erneut zu kalkulieren. Der größte zeitliche Aufwand versteckt sich hinter Punkt Sechs, der erneut auf den „Quadrieren und Multiplizieren“-Algorithmus mit der Blakley-Multiplikation aufbaut. Das binäre modulare Potenzieren arbeitet jedoch auf Grund der Zusatzberechnungen in einer Form, die um die halbe Bitbreite reduziert ist.

In der im folgenden Abschnitt diskutierten 128 Bit Implementierung wurden die Schritte Drei bis Acht umgesetzt und auf Seitenkanäle analysiert.

6.2.1 Rechenzeitanalyse

Die Berechnung der modularen Inversen q_{inv} (Schritt Drei in der Ablaufreihenfolge des gesamten Algorithmus) hängt zeitlich nur von den beiden Primzahlen p und q ab und bleibt daher für einen unveränderten Schlüssel konstant. Die Dauer des Rechenvorgangs trägt somit keine direkte Seitenkanalinformation.

Für den Schritt Vier und Fünf benötigt CRT-RSA eine Modulo-Berechnung. Ein Algorithmus für Modulo-Operationen, der nicht versucht Seitenkanalinformationen zu verschleiern, sagt durch die Dauer seiner Ausführung viel über seine Operatoren aus. Hier gilt: Je höher die Wertigkeit dieser Daten, desto länger benötigt der Reduziervorgang.

Dieser Angriff kommt für die Subschlüsselberechnung d_q und d_p nicht in Frage, in der eine Modulo-Operation den Schlüssel reduziert. Dies hat den Grund, dass die Operatoren eine konstante Länge besitzen und nicht durch den Angreifer vorgegeben werden können. Außerdem weist das Ergebnis keine direkte Abhängigkeit zum verschlüsselten Wert auf und kann so vom untersuchten Algorithmus schon im Vorhinein vollzogen werden. Dies schränkt eine Beobachtbarkeit stark ein. Hingegen stellt sich Schritt Fünf, welcher die Eingangsdaten reduziert, als besonders geeignete Quelle für Seitenkanalinformationen heraus. Die Verknüpfung der Eingangsdaten mit der Ausführungszeit lässt Rückschlüsse auf die beiden Primzahlen p und q zu. Mit der Annahme, dass es dem Angreifer gelingt, die Eingangsvektoren für die Verschlüsselung vorzugeben, kann durch das Setzen eines kleinen und großen Werts im Verlustleistungsprofil der zugehörige Abschnitt gefunden werden.

Da nun der Abschnitt für die Berechnung von Schritt Fünf bekannt ist, kann durch die Vorgabe der Eingangsdaten und iteratives Probieren versucht werden, einen Wert zwischen den beiden Zahlen p und q zu finden. Ein Wert dafür gilt als gefunden, wenn die Ausführungszeit für beide Modulo-Berechnungen einen Taktzyklus länger benötigt, als für besonders kleine Eingangsdaten. Durch weiteres schrittweises Vorgehen nähert sich der Angreifer vom höchstgesetzten Bit dem genauen Zahlenwert an, bis nur noch das niederwertigste Bit angibt, wie lange die Modulo-Komponente arbeitet. Mit dieser Methode lassen sich die beiden gesuchten Primzahlen aus der Rechenzeitanalyse extrahieren.

Um zu zeigen, dass es sich bei dieser Verwundbarkeit nicht um einen Spezialfall einer Implementierung handelt, wurden drei verschiedene Modulo-Implementierungen getestet und erfolgreich attackiert.

Abbildung 6.6 zeigt die unterschiedliche Ausführungszeit einer dieser Modulo-Implementierungen. Bei einer Taktperiode von 100 ns erkennt man im oberen Teilbild die Bearbeitung von reduzierbaren Eingangsdaten, die nur gering oberhalb der kleineren Primzahl liegen. Darunter sieht man dieselbe Modulo-Operation, angewandt auf andere Eingangsdaten, die auf Grund ihrer Größe keine Reduktion benötigt und deshalb einen Takt früher beendet werden kann.

Anhand des öffentlich bekannten Schlüssels (e, N) lassen sich die beiden Primzahlen auf ihre Korrektheit prüfen. Dazu steht die Formel $N = p \cdot q$ zur Verfügung. Trifft dies zu, errechnet sich der private Schlüssel mit der eulerschen Funktion $\varphi(n) = (p-1) \cdot (q-1)$ und der multiplikativen Inverse zu

$$d = \frac{1}{e} \mod \varphi(n) \ . \tag{6.4}$$

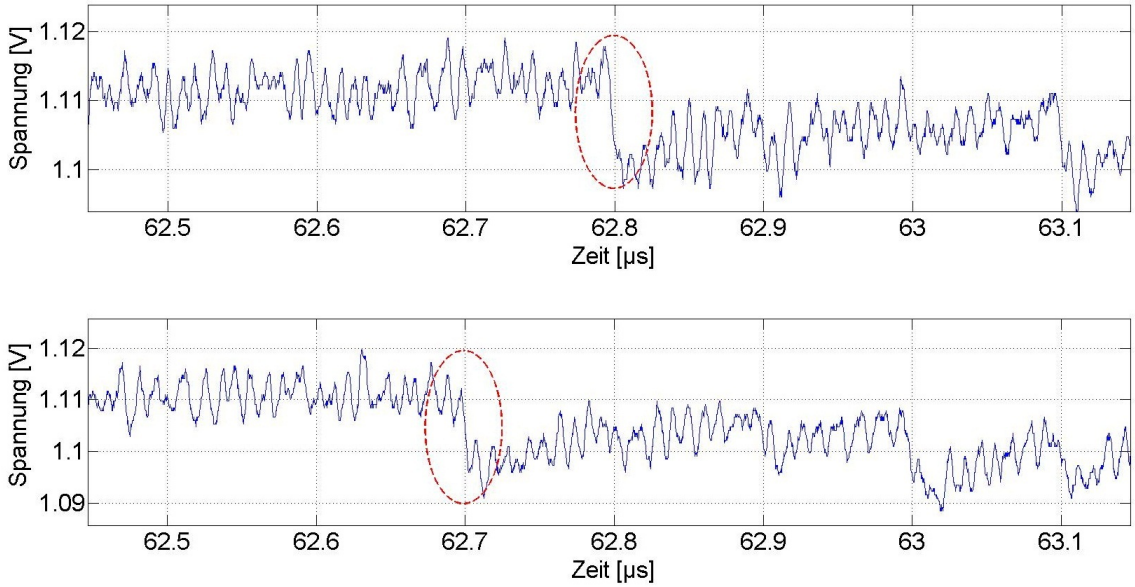

Abbildung 6.6: Unterschiedliche Ausführungszeit einer Modulo-Operation

Diese beschriebene Seitenkanalattacke nimmt an, dass ein Angreifer in der Lage ist, die verschlüsselte Nachricht vorzugeben. Ist dies nicht der Fall und Eingangsdaten können nur beobachtet werden, stehen auch statistische Mittel zur Verfügung, um die beiden Zahlen p und q zu eruieren. Dazu muss der Modulo-Algorithmus bekannt sein und die beobachteten verschlüsselten Eingangswerte (C) sich gleichmäßig über die volle Bandbreite verteilen. Ein statistisches Modell für die primitivste Modulo-Operation, die solange den Modulus p subtrahiert, bis der Operand reduziert ist, wird im Weiteren als Beispiel näher betrachtet:

Die Eingangsdaten konvergieren dank der Gleichverteilung zum statistischen Mittelwert $\frac{N}{2} = \frac{p \cdot q}{2}$. Der Modulus p begrenzt die Eingangsdaten C für die Verschlüsselung auf seine Größe und benötigt dafür T Taktschritte. Diese Anzahl gibt in diesem Modulo-Vorgang die Anzahl der Subtraktionen wieder und grenzt den Wert C ab:

$$T \cdot p \leq C < (T+1) \cdot p \tag{6.5}$$

Die Division dieser Ungleichung durch T und die Mittelung von mehreren Messungen mit unterschiedlichen Eingangsdaten lässt für große Werte von T erkennen, dass $\frac{C}{T}$ gegen p konvergiert. Sollte kein Zugriff auf den öffentlichen Modulus vorliegen, lässt sich dieser mit Hilfe des Mittelwerts der verschlüsselten Nachricht C abschätzen. Dieser Mittelwert dividiert durch p ergibt $\frac{q}{2}$ und somit die zweite Primzahl q. Die Gültigkeit dieses Modells konnte mit einem Simulationsmodell in C verifiziert werden.

Für andere Algorithmen, die eine Modulo-Berechnung in mehrere Taktzyklen umsetzen, ergeben sich andere Schranken. Es gilt: Je schneller der Algorithmus die Zahl reduzieren kann, desto größer ist der Abstand zwischen den Schranken und desto mehr Operationen sind für dieselbe Genauigkeit zu beobachten.

6.2.2 Simple Power Analysis

Die Grundlage für die Extremwertanalyse, dass die Multiplikation mit einer kleinen Zahl zu einer geringen Verlustleistung führt, ist im Ablaufschritt Sechs des Rechenvorgangs mit der vorherigen Implementierung in Abschnitt 6.1 ident. Extremwerte können dazu verwendet werden, den Exponenten im binären modularen Potenzieren zu detektieren, den Abschnitt im gemessenen Leistungsprofil zu finden und von anderen Vorgängen abzugrenzen.

Das Attackieren der beiden Subschlüssel d_q und d_p kann dabei mit einem kleinen Extremwert und einer adaptierten $N-1$ Extremwertmethode erfolgen. Die Adaption führt aufgrund des geänderten Modulus im Exponenzierer zu den beiden speziellen Nachrichten $p-1$ beziehungsweise $q-1$. Damit dies funktioniert, müssen die beiden Zahlen p und q zuvor mit der Rechenzeitanalyse bekannt gemacht werden. Anders verhält es sich mit kleinen Extremwerten. Hier kann auch ohne das Wissen der Primzahlen der Subschlüssel gefunden werden.

Das Finden der beiden Subschlüssel d_q und d_p hat nur dann Sinn, wenn der öffentliche Schlüssel d nicht bekannt ist, da man in diesem Fall nicht auf den geheimen Schlüssel e rückrechnen kann. Mit den beiden Subschlüsseln und den Primzahlen p und q aus der Rechenzeitanalyse stehen alle Informationen zur Verfügung, um eine Nachricht zu entschlüsseln, auch wenn keine Möglichkeit besteht, auf den verwendeten privaten Schlüssel rückzuschließen. Die Rechenschritte Drei und Fünf bis Acht zeigen diese Vorgangsweise ohne der Verwendung der Verschlüsselungsparameter N, e und d.

6.2.3 Differential Power Analysis

Von der Differential Power Analysis kann im sechsten Rechenschritt Gebrauch gemacht werden, um die Subschlüssel zu erlangen, wenn nur die Eingangsdaten bekannt, aber nicht von extern modifizierbar sind und die beiden Primzahlen p und q bereits detektiert wurden. Die genaue Vorgangsweise mit einem Simulationsmodell dazu findet sich im Kapitel 6.1 im Abschnitt „Differential Power Analysis“. Wo zuvor nur ein Rechenvorgang $M = C^d \mod N$ attackiert wurde, müssen nun zwei Rechenvorgänge $M_p = C_p^{d_p} \mod p$ und $M_q = C_q^{d_q} \mod q$ angegriffen werden.

Auch ein Angriff mit einem Verlustleistungsmodell auf die Reduzierung der Eingangsdaten C (Schritt Fünf) ist denkbar. Betrachtet man jedoch den Umstand, dass es eine Vielzahl an unterschiedlichen Algorithmen für die Modulo-Berechnung gibt, die zu vielen möglichen Leistungsmodellen führen, hat eine Analyse der Ausführungszeit ein höheres Potential als die Beobachtung von internen Vorgängen.

6.3 „Quadrieren und Multiplizieren“-„Montgomery“-Algorithmus

Dieses Kapitel behandelt einen „Quadrieren und Multiplizieren“-Algorithmus, der als Recheneinheiten die Montgomery-Multiplikation verwendet. Dieser Algorithmus wandelt zuerst die verschlüsselte Nachricht C in den sogenannten Montgomery-Raum um, wodurch die Modulo-Subtraktion als Shift-Operation ausgeführt wird. Diesem Umstand zufolge verbraucht diese Implementierung im Vergleich zu der Blakley-Multiplikation weniger Hardwareressourcen im FPGA. Tabelle 6.1 zeigt diese Unterschiede, welche in der Synthese-Übersicht mit dem Tool „ISE Design

Suite 14.7“ für eine 128 Bit Implementierung sichtbar sind. Die Auflistung zeigt die starke Differenz zwischen den benötigten Lookup-Tabellen (LUT) und Multiplexern für Überträge (MUXCY) in der Umsetzung, welche durch die Vereinfachung der Reduktion im Montgomery-Raum entsteht.

Tabelle 6.1: Unterschied der Hardwareressourcen zwischen Montgomery- und Blakley-Implementierung

Ressourcen	Montgomery-Multiplikation	Blakley-Multiplikation
Slice Registers	2.339	2.725
Slice LUTs	2.658	6.072
MUXCY	904	2.856

Der Ablauf für diese Implementierungsvariante beginnt zuerst mit einer Transformation des Eingangswerts C und dem initialen Einselement in den Montgomery-Raum. Die beiden transformierten Werte ergeben sich aus dem Wert R, der die nächstgrößte Zweierpotenz zu N darstellt, zu

$$\overline{C} = C \cdot R \mod N \tag{6.6}$$

und

$$\overline{1} = 1 \cdot R \mod N. \tag{6.7}$$

Die Operanden dieser Berechnung beinhalten bei Kenntnis der Eingangsdaten und des öffentlichen Schlüssels (e, N) keine relevanten Informationen für eine Seitenkanalanalyse und werden deshalb nicht weiter betrachtet. Außerdem verhält sich die Spannungsvariation in diesem gemessenen Abschnitt relativ unauffällig, da es sich bei der Zahl R um eine Zweierpotenz handelt. Wesentlich mehr Informationen beinhaltet das - in Abhängigkeit vom geheimen Schlüssel ausgeführte - Quadrieren und Multiplizieren. Je nach gesetztem Bit verändern sich die Operanden der Multiplikation im Montgomery-Raum. Angesichts dessen, dass es sich hier nicht um eine gewöhnliche Operation handelt, wird das Multiplikationszeichen in diesem Raum im Weiteren mit einem Kreis ($\circ$) symbolisiert. Der Aufbau dieses Rechenschritts, welcher für die Seitenkanalanalyse essenziell ist, wird in Form des Implementierungscodes in VHDL im Abschnitt „Differential Power Analysis“ näher gebracht. Nach dem Potenzieren im Montgomery-Raum hilft die Multiplikation mit dem Einselement im Ursprungsraum die Rückwandlung ohne zusätzliche logische Transformationseinheit zu verwirklichen. Eine mathematische Betrachtung zu dieser Rechenmethode findet sich im Kapitel 3.2.3.2.

6.3.1 Rechenzeitanalysis

Die Ausführungszeit des Algorithmus hängt nicht, wie im „Quadrieren und Multiplizieren“-„Blakley“-Algorithmus, von der am Eingang anliegenden Nachricht ab. Dennoch kann aus der Ausführungszeit der Verschlüsselung auf die Anzahl der gesetzten Bits - das Hamming-Gewicht - Rückschluss gezogen werden. Das Hamming-Gewicht des Schlüssels gibt dabei die Anzahl der benötigten zusätzlichen Multiplikationen vor, die zu einer längeren, messbaren Berechnungszeit der Verschlüsselung führen. Dieser Angriffsvektor spielt bei konstantem Schlüssel nur eine unterstützende Rolle für eine weitere Seitenkanalanalyse. Um die Schlüssellänge zu validieren oder den Schlüssel einzuschränken, kann jedoch das Hamming-Gewicht herangezogen werden.

6.3.2 Simple Power Analysis

Um durch eine Messaufnahme direkt den Schlüssel oder andere Informationen aus dem Verschlüsselungsvorgang zu gewinnen, wurde mittels Extremwerte versucht auf diese rückzuschließen. Dazu kamen wieder die zwei besonderen Werte $N-1$ und ein möglichst kleiner Wert oberhalb von Eins für die Verschlüsselung zum Einsatz (Kapitel 6.1.2). Diese Technik über die Extremwertanalyse, wie sie bereits für die RSA-Implementierung „Quadrieren und Multiplizieren“-„Blakley“-Algorithmus verwendet wurde, hat andere Auswirkungen im Montgomery-Raum, welche im Folgenden beschrieben werden sollen.

Es stellt sich dabei heraus, dass die Transformation eines kleinen Werts im Ursprungsraum zu einem zufälligen, mit hoher Wahrscheinlichkeit relativ großen Wert im transformierten Zustand führt und deswegen keine Informationen preisgibt. Es kann jedoch ein Wert im Ursprungsraum gefunden werden, der im Montgomery-Raum zu einer kleinen Zahl X führt. Dieser Wert C berechnet sich aus $C = X \circ 1$, was der Rücktransformation einer beliebigen kleinen Zahl X aus dem Montgomery-Raum entspricht. Für diese Transformation benötigt ein Angreifer lediglich die Transformationszahl R und den öffentlich bekannten Modulus N. Die Zahl R repräsentiert die nächstgrößere Zweierpotenz zu N und ist deshalb einfach berechenbar.

Ein Tiefpassfilter erster Ordnung mit einer Grenzfrequenz von 2 MHz hilft die Informationen in einer Messaufnahme bei einer Taktfrequenz des FPGAs von 10 MHz erkennbar darzustellen. Die Grenzfrequenz des Tiefpasses liegt unter der Taktfrequenz, da die Montgomery-Operation über mehrere Taktzyklen arbeitet und somit den lokalen Gleichanteil für die Detektion sichtbar macht. Das gefilterte Messsignal, welches die Unterscheidung möglich macht, findet sich in Abbildung 6.7 wieder. Eine Multiplikation des Zwischenergebnisses mit einer kleinen Zahl im Montgomery-Raum wird in der Grafik mit dem Buchstaben „M“ und das Quadrieren des Zwischenergebnisses mit „Q“ gekennzeichnet.

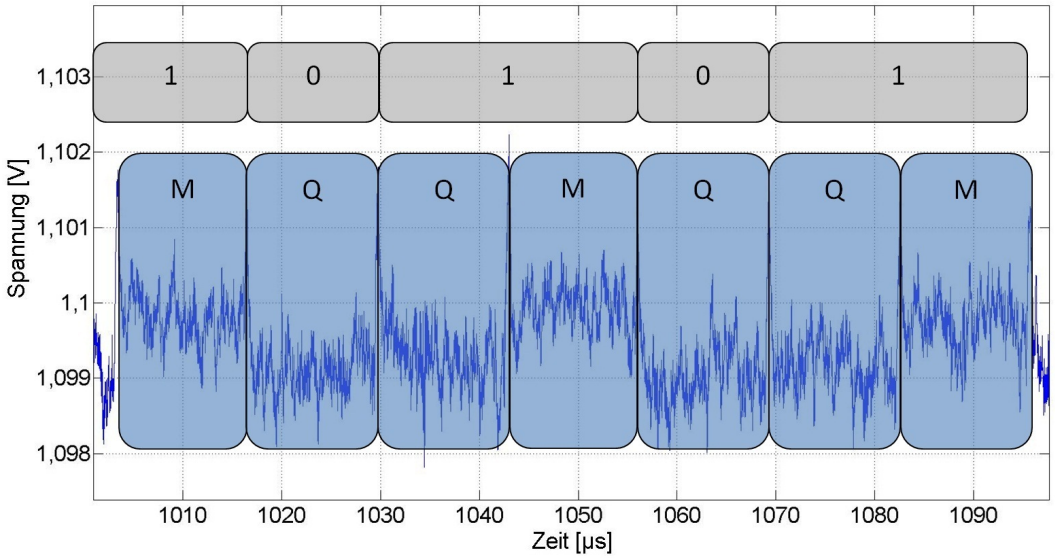

Abbildung 6.7: Seitenkanalangriff mit kleinem Extremwert im Montgomery-Raum

Ein weiterer wichtiger Unterschied liegt in der Analyse des $N-1$ Angriffs. Im Montgomery-Raum ist es ebenfalls möglich bei der Vorgabe dieses Extremwerts die Berechnungen auf drei verschiedene Vorgänge zu reduzieren:

1. Quadrieren des Einselements ($\overline{1} \circ \overline{1} = \overline{1}$)
2. Das Montgomery-Produkt des Extremwerts mit dem Einselement ($\overline{1} \circ \overline{(N-1)} = \overline{(N-1)}$)
3. Quadrieren des Extremwerts ($\overline{(N-1)} \circ \overline{(N-1)} = \overline{1}$)

Durch die Transformation nehmen die Extremwerte im Ursprungsraum ebenfalls zufällige Werte im Montgomery-Raum an. Aus diesem Grund erzeugt beispielsweise das Quadrieren des Einselements eine unauffällige und nicht minimale Verlustleistung. In der getesteten 128 Bit Implementierung konnten diese drei Operationen auseinandergehalten werden. Diese Unterscheidung schafft die Fähigkeit, den gesuchten Schlüssel im Leistungsprofil zu enttarnen.

Damit es gelingt, die drei verschiedene Abschnitte der richtigen Operation zuzuordnen, hilft es die Vorschriften und Eigenheiten des „Quadrieren und Multiplizieren"-Algorithmus zur Hilfe zu nehmen. Bei der ersten Operation handelt es sich für gewöhnlich um die Multiplikation $\overline{1} \circ \overline{(N-1)}$. Die Implementierung kann auf diese Operation verzichten, indem sie das erste Zwischenergebnis direkt mit $\overline{(N-1)}$ initialisiert. In der darauf anschließenden Operation, welche immer dieser Multiplikation folgt, findet der Vorgang des Quadrierens des Extremwerts statt ($\overline{(N-1)} \circ \overline{(N-1)}$). Des Weiteren besteht die Möglichkeit, aus der Anzahl der verschiedenen Montgomery-Multiplikationen auf die Häufigkeit der drei unterschiedlichen Berechnungsvorgänge zu schließen. In einem k Bit Verschlüsselungsalgorithmus dieser Form gibt es für die Anzahl z der gemessenen Operationen $z-k$ Multiplikationen ($\overline{1} \circ \overline{(N-1)}$) und $z-k$ Quadrier-Operationen nach einer Multiplikation ($\overline{(N-1)} \circ \overline{(N-1)}$). Wenn am Anfang das erste Zwischenergebnis bereits mit $\overline{(N-1)}$ initialisiert wurde, wird eine Multiplikation weniger benötigt. Zu der letzten Operation, die durch ihre Auftrittshäufigkeit ($2 \cdot k - z$) die gesamte Summe ergänzt, gehört das Quadrieren des Einselements.

Eine 32 Bit Verschlüsselungsimplementierung mit „Quadrieren und Multiplizieren"-Algorithmus, die mit 40 Operationseinheiten arbeitet, hat demnach circa acht Multiplikationen und acht darauffolgende Quadrier-Operationen. Die etwa 24 restlichen Operationseinheiten werden für das Produkt der beiden Einselemente verwendet.

6.3.3 Differential Power Analysis

Die Simple Power Analysis macht durch die Messaufnahme in Folge von vorgegebenen Extremwerten markante Leistungsprofilabschnitte sichtbar. Sollten die verschlüsselten Eingangsdaten nicht verändert werden können aber weiterhin beobachtbar sein, kann dieser Angriffsvektor helfen, den privaten Schlüssel zu erlangen. Eine in einem C-Programm entwickelte Simulation hilft - ähnlich wie im Unterkapitel 6.1.3 - ein Leistungsmodell für den nächst möglichen Operationsschritt (Multiplizieren oder Quadrieren) zu erstellen und den Schlüssel im Profil iterativ bitweise nachzubauen. Um solch ein Modell zu erstellen, benötigt der Angreifer Detailinformationen zur Implementierung, wie den dazugehörigen VHDL-Code, oder eine Selektion von plausiblen Implementierungsvarianten. Im Folgenden wird der VHDL-Code 6.2 einer Montgomery-Implementierung genauer untersucht:

VHDL-Code 6.2: Montgomery-Multiplikation

```
1 p_MMcell: process(clk_i) variable P
2 : UNSIGNED(bitwidth +1 downto 0):=(OTHERS=>'0'); begin
3  if rising_edge(clk_i) then
4   if rst_i='1' then
```

```
5       s_busy <='0';
6     else
7      if (s_ld='1' and s_busy='0') or (s_busy='1') then
8       if s_ld='1' then
9        s_busy <='1';
10       counter <=0;
11       P:=(OTHERS=>'0');
12       bb<=UNSIGNED(state2_i); -- Laden der Eingangsoperatoren
13       s_state1<=state1_i;
14      else
15       if counter = bitwidth then -- Letzte Runde
16        if P>= ( "00" & UNSIGNED(modul_i)) then -- Finales Reduzieren
17         P:=P-( "00" & UNSIGNED(modul_i)) ;
18        end if;
19        state_o<=std_logic_vector(P((bitwidth -1) downto 0));
20        s_busy <='0';
21       else
22        if s_state1(counter) = '1' then
23         P:=P+("00"& bb); -- Binaeres Multiplizieren (P=P+a(counter)*b)
24        end if;
25        if P(0) = '1' then --Wenn P ungerade ist addiere den Modulus, damit
26            -- kein Informationsverlust bei der Division durch 2 entsteht
27         P:=P+("00"& UNSIGNED(modul_i));
28        end if;
29        P:= '0' & P(bitwidth + 1 downto 1); -- Division durch 2 (Shift)
30        counter<=counter+1;
31       end if;
32      end if;
33     end if;
34    end if;
35   end if;
36 end process;
```

Nach dem Laden der Eingangsoperatoren in den Zeilen 13 und 14 arbeitet die Montgomery-Multiplikation in den weiteren Taktzyklen zwischen den Codezeilen 23 und 31. In diesem Codestück kann man interne Vorgänge ablesen, die zu messbaren Leistungsaufnahmeänderungen am FPGA führen. Die optionalen Statements, wie das binäre Multiplizieren und die Addition des Modulus wenn P ungerade ist[2], sind dafür zuständig. Die Division durch die Zahl Zwei in der Form einer Shift-Operation wird in jeder Runde ausgeführt. Der Code unterscheidet deshalb vier unterschiedliche Ausführungspfade:

1. Shift
2. Binäres Multiplizieren, Shift
3. Addition Modulus, Shift
4. Binäres Multiplizieren, Addition Modulus, Shift

Das Modell verwendet für die Gewichtung nicht die Ausführungspfade selbst, sondern den Wechsel zwischen diesen unterschiedlichen Möglichkeiten des Programmablaufs. Aus diesem Grund müssen 16 Gewichte für die Pfadänderung definiert werden. Der stärkste Spannungseinbruch tritt im

[2]Die Addition führt zu einer geraden Zahl, da die Vorbedingung für die Montgomery-Multiplikation ein ungerades Modulus vorschreibt.

Simulationsmodell zwischen den Ausführungszweigen Eins und Vier ein. Der geringste Aufwand entsteht, wenn derselbe Pfad im nächsten Taktzyklus erneut zur Aktivierung kommt.

Vergleicht man den Spannungseinbruch der Blakley-Multiplikation (Kapitel 6.1) mit jener der Montgomery-Multiplikation, sieht man einen deutlichen Unterschied in der Variation der Spannung während des Verschlüsselungsvorgangs. Der in diesem Abschnitt betrachtete Algorithmus arbeitet durch den niedrigeren Logikverbrauch effektiver und hat auch einen möglichen Ausführungspfad weniger. Diese Umstände führen zu einem deutlich geringeren Signal-Rausch-Verhältnis (SNR), welches in der Messung beobachtet wurde. Um dem entgegen zu wirken, wurden mehrere Messungen gemittelt und eine 254 Bit sichere Verschlüsselungsimplementierung, anstatt der bisherigen 128 Bit Implementierungen gewählt. Diese zwei Maßnahmen produzieren ein deutlich höheres SNR der messbaren Effekte und somit deutlich bessere Ergebnisse im Abgleich mit dem erstellten Leistungsmodell.

Mit einer automatisierten Seitenkanalanalyse, die als Kernkomponente das Programm MATLAB® verwendet, berechnet sich der Mittelwert zu jedem Abtastzeitpunkt aus 200 Messaufnahmen derselben Multiplikationsoperation. Vergleicht man diesen mit den zwei möglichen Rechenoperationen (Multiplizieren oder Quadrieren) aus der Simulation, so lässt sich anhand der Korrelation feststellen, welches Modell eine bessere Übereinstimmung aufweist. Abbildung 6.8 bildet das richtige Modell mit dem dazugehörigem Spannungsverlauf ab. Die Korrelation mit diesem Modell und der falschen Modellannahme zeigt Grafik 6.9. Die Korrelation mit dem übereinstimmenden Modell ist durch die spitze Flanke in der Mitte identifizierbar.

Aus der positiven Übereinstimmung ist ersichtlich, dass durch die Erhöhung der Verschlüsselungsbreite und mehreren Messungen auch ein effektiver Algorithmus wie die Montgomery-Multiplikation angreifbar ist. Da die Vor- und Nachteile dieses Angriffsvektors bereits in Abschnitt 6.1.3 ausgiebig zur Diskussion standen und diese hier analog gelten, soll hiermit nochmals darauf verwiesen werden.

6.4 „Chinesischer Restsatz“-„Montgomery“-Algorithmus

Die letzte Implementierung geht von einer Architektur aus, die die zwei Primzahlen der Schlüsselgenerierung für das chinesische Restsatz Theorem (CRT) verwendet und dazu den modularen Potenzierer aus dem vorherigen Kapitel einbindet. Es handelt sich hier um einen reinen Entschlüsselungsalgorithmus, der versucht, die Bit-Breite der Grundoperationen, wie Addition und Subtraktion, zu minimieren. Der Angriff auf den CRT relevanten Teil, insbesondere mit Hilfe der Rechenzeitanalyse, kann Kapitel 6.2 entnommen werden, da sich hier keine konzeptuellen Änderungen ergeben. Die Anwendung der „Simple Power Analysis“ und der „Differential Power Analysis“ führen durch den Einsatz von CRT und der Montgomery-Multiplikation, zu Änderungen, die in diesem Abschnitt zur Diskussion stehen.

6.4.1 Simple Power Analysis

Wie bereits angedeutet, funktioniert der Angriff über Extremwerte zu unterschiedlichen Betrachtungsweisen und Angriffsflächen im Vergleich zu der SPA für einen herkömmlichen Montgomery-Multiplikation-Algorithmus. Um die beiden Extremwerte $p - 1$ und $q - 1$ am Montgomery-Exponenzierer (Algorithmus 3.5 im Kapitel 3.2.3.2) anzulegen, müssen diese zwei Werte zuvor

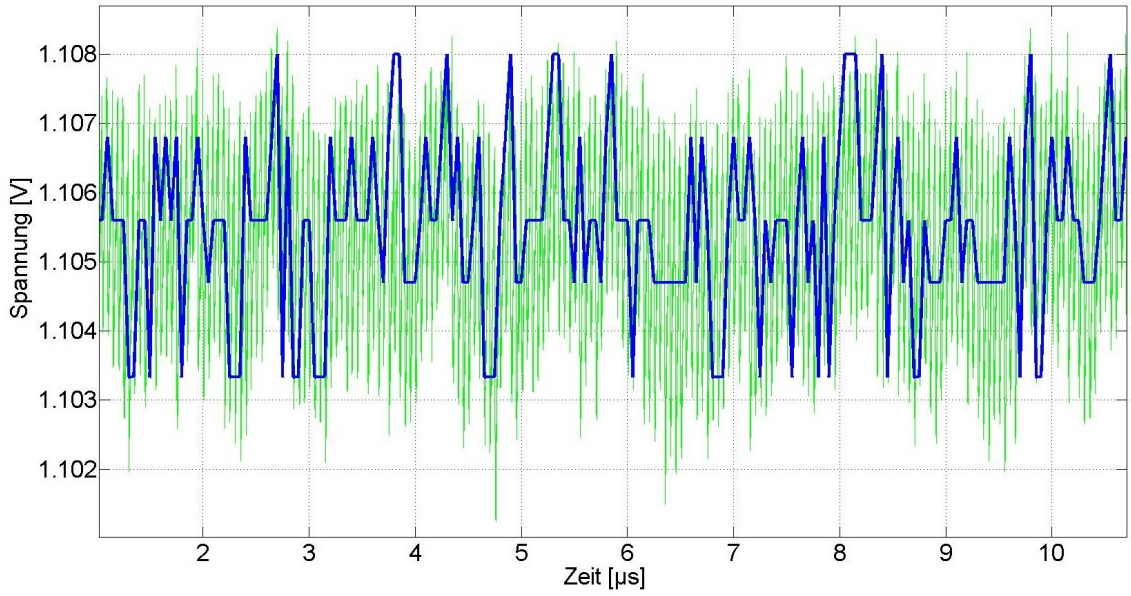

Abbildung 6.8: Überlagerung des realen Spannungseinbruchs durch die Montgomery-Multiplikation in Grün mit dem in Blau eingezeichneten, generierten Simulationsmodell

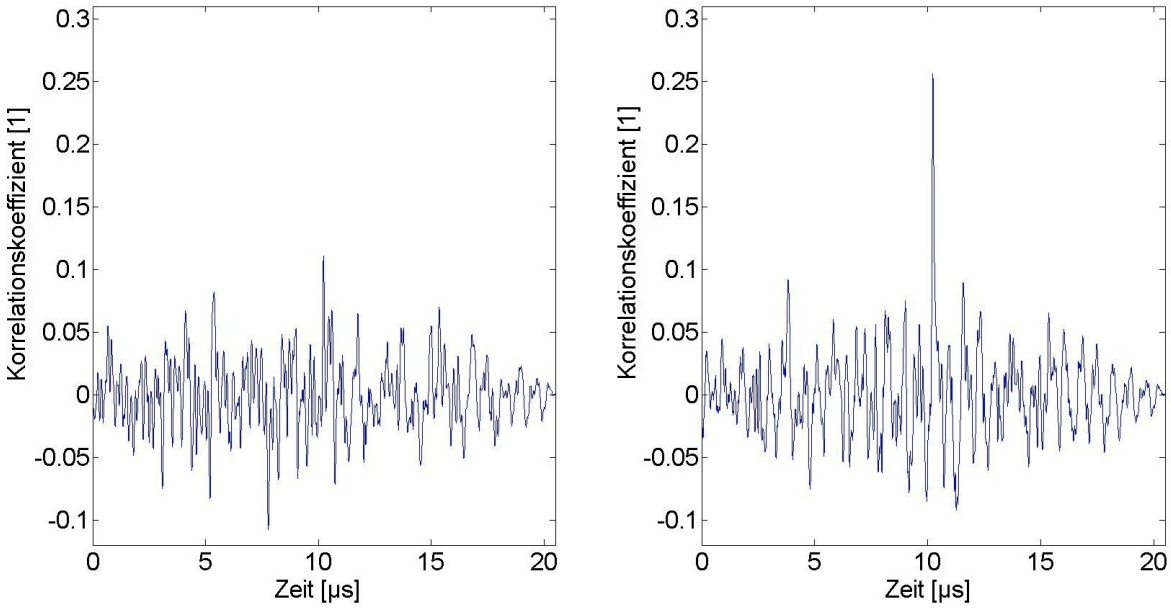

Abbildung 6.9: Kreuzkorrelation (Montgomery) - Links: falsches Modell - Rechts: richtiges Modell

aus der Rechenzeitanalyse extrahiert werden. Dies gelingt vor allem dann, wenn die vorangestellte Modulo-Operation in seiner Implementierung bekannt ist oder in der Ausführungszeit einen

Zeitunterschied zwischen oberhalb beziehungsweise unterhalb des Modulus verursacht.

Ein anderes Angriffsszenario bieten kleine vorgegebene Werte, da die Multiplikation im Montgomery-Raum eines kleinen mit einem großen Werts und das Quadrieren zweier größerer Werte einen Unterschied in der Seitenkanalmessung verursacht. Im Vergleich zu einem klassischen Extremwertangriff mit kleinen Eingangswerten in Kapitel 6.1.2, muss hier eine Nachricht angelegt werden, die erst im Montgomery-Raum transformiert einen kleinen Wert ergibt. Für die Berechnung dieses Werts X im Ursprungsraum wird die Montgomery-Multiplikation in der Form

$$C = X \circ 1 = X \cdot 1 \cdot R^{-1} \mod N \tag{6.8}$$

ausgenutzt. Der Transformationswert R, welcher die nächsthöhere Zweierpotenz zum jeweiligen angegriffenen Modulus p beziehungsweise q der modularen Potenzierfunktionen widerspiegelt, muss im Vergleich zu der vorherigen Implementierung in Kapitel 6.3 geschätzt werden. Diese beiden Primzahlen besitzen aus Sicherheitsgründen meist dieselbe Größe und lassen deshalb den Transformationswert relativ gut bestimmen. Für einen 128 Bit Montgomery-Exponenzierer gibt es theoretisch nur 128 Möglichkeiten für Zweierpotenzen, die für die Transformation sinnvoll wären.

6.4.2 Differential Power Analysis

Das erstellte Simulationsmodell für den Korrelationsangriff aus dem Kapitel 6.3 kann nur dann verwendet werden, wenn die beiden Primzahlen p und q, die den Modulus des binären modularen Potenzierens vorgeben, bekannt sind. Solch ein Angriff macht nur dann Sinn, wenn der öffentliche Schlüssel e nicht zur Verfügung steht. Da sonst aus dem mathematischen Zusammenhang zwischen den beiden Primzahlen und dem öffentlichen Exponenten e der gesuchte Schlüssel berechnet werden kann.

6.5 Verifikation der Ergebnisse

Dieser Unterabschnitt behandelt die Verifikation der im Kapitel 6 verwendeten Angriffsmethoden, wobei der Seitenkanal abhängig vom Angriffsvektor geprüft wurde.

Der praktische Seitenkanalangriff auf unterschiedliche Implementierungen mit Hilfe von SPA verifiziert einerseits die Detektionsmethoden mit speziellen Nachrichten und zeigt andererseits, wie aus diesem Muster der damit verbundene Schlüssel gewonnen werden kann. Dazu kommen neben den mathematischen Schlussfolgerungen für Extremwerte hauptsächlich empirische Methoden zum Einsatz, die auf verschiedenen Parametern der Verschlüsselungen basieren.

Für einzelne Messaufnahmen des Spannungseinbruchs an der FPGA-Versorgung wurden zehn verschiedene Schlüssel generiert, auf die jeweils mehrere kleine Werte und der $N-1$ Extremwert für die Rekonstruktion des gesuchten Schlüssels erfolgreich eingesetzt wurden. Die deutlichen Unterscheidungsmerkmale in der Messung, sichtbar in Abbildung 6.2, weisen auf einen sehr stabilen Angriffsvektor hin.

Anhand der zeitlichen Analyse wurde gezeigt, wie durch die Messung der Ausführungszeit Informationen aus einem Verschlüsselungsvorgang extrahiert werden können. Als besonders anfällig stellte sich der CRT-RSA-Algorithmus in den Untersuchungen heraus, der die zeitliche Variation der Modulo-Operation nutzt. Die Analyse von drei unterschiedlichen Modulo-Implementierungen,

die alle auf dieselbe Weise attackiert wurden, ließ erkennen, dass es sich nicht um ein spezielles Verhalten einer Modulo-Implementierung handelt. Zusätzlich wurden neben den verschiedenen Umsetzungen der Modulo-Operation fünf Erzeugerprimzahlpaare für den Angriffsvorgang untersucht. Der Erfolg aller durchgeführten Angriffe demonstrierte die Stärke des Angriffsvektors und belegte die Gültigkeit der Algorithmen-Detektionsmethode „Variation der Eingangsdaten“.

Auf Seitenkanalangriffe, die mit Autokorrelation und Verlustleistungssimulationsmodellen arbeiten, wurden ebenfalls zehn verschiedene Schlüssel angewandt, wobei die Korrelation nur auf Teilen des privaten Schlüssels ausgeführt wurde. Einige wenige Messungen konnten der gesuchten Operation nicht zugeordnet werden. Was jedoch mit dem Folgefehlermodell (beschrieben in Kapitel 6.1) im nächsten untersuchten Bit korrigiert werden konnte. Die Attacke benutzt dazu mehrere randomisiert generierte Nachrichten, um eine zufällige Übereinstimmung der Messung mit der Simulation auszuschließen. Die Mittelung von 200 Messungen, die das Rauschen unterdrücken soll, beweist, dass der gemessenen Effekt ein eindeutiges, immer wiederkehrendes Muster aufweist. Verwendet man jedoch einen abgewandelten VHDL-Code als Basis für das Simulationsmodell, kann eine Abweichung und damit keine Übereinstimmung in der Korrelation entstehen. Deshalb ist diese Detektionsmethode nur mit einer großen Modelldatenbank oder genügend Detailinformationen zur Implementierung sinnvoll.

7 Analyse des Messumfeldes und Charakteristiken schaltender Logik

Um den Seitenkanal „Verlustleistung“ auf einem FPGA-Entwicklungsboard zu messen, muss zuerst ein funktionierender Messaufbau vorbereitet werden, damit interne Verarbeitungsvorgänge beobachtet werden können. Verschiedene Ansätze dazu und genauere Informationen zu diesen Messaufbauten sind Thema dieses Abschnitts. Außerdem geben erste Untersuchungen der Messergebnisse Einblick in die Messbarkeit von Seitenkanalsignalen.

7.1 Messumfeld

Um möglichst genaue Seitenkanalinformationen anhand der Verlustleistung zu erhalten, ist eine gut eingestellte Messschaltung unumgänglich. Dabei wird in dieser Arbeit die Änderung der Stromaufnahme des FPGAs beobachtet, die im Zusammenhang mit der Leistungsvariation des zu untersuchenden Devices steht.

Es gibt verschiedene Möglichkeiten Messwerte für die Verlustleistung- beziehungsweise die Stromverbrauchsänderungen aufzunehmen. Als Messgerät kann dazu ein digitales Multimeter oder ein Oszilloskop dienen. Messgeräte mit einer Samplefrequenz ab 1 GS/s und einer Mindestbandbreite von circa 500 MHz stellen sich in dieser Arbeit als ausreichend dar. Diese Anforderungen sind jedoch stark von der Technologie und der Taktfrequenz des Zielsystems abhängig.

Mithilfe von Strommesszangen kann auf einen zusätzlichen Messwiderstand verzichtet werden, wodurch eine Messabweichung über den Spannungsabfall am Widerstand die Messgröße nicht beeinflusst. Das führt jedoch dazu, dass nur Änderungen im Stromverbrauch und keine statischen Strommessungen aufgenommen werden können. Um Spannungen über Shunt-Widerstände zu messen, setzt man oft „Single Ended“- oder differentielle Tastköpfe ein. „Single Ended“-Tastköpfe sind in der Anschaffung zwar wesentlich billiger, können jedoch nur Ground als Bezugsspannung annehmen.

Die verschiedenen Messschaltungskonzepte können der Abbildung 7.1 entnommen werden. Der Messwiderstand im linken Teilbild wird nahe der Groundleitung positioniert. Dem Vorteil, dass dieser Aufbau einfach realisierbar ist, stehen einige Nachteile gegenüber. Interne Glättungskondensatoren, die eine stabile Versorgung gewährleisten, erschweren einerseits das Messen des Seitenkanals. Zum anderen existieren auf Schaltungsplatinen oft andere elektronische Bauteile

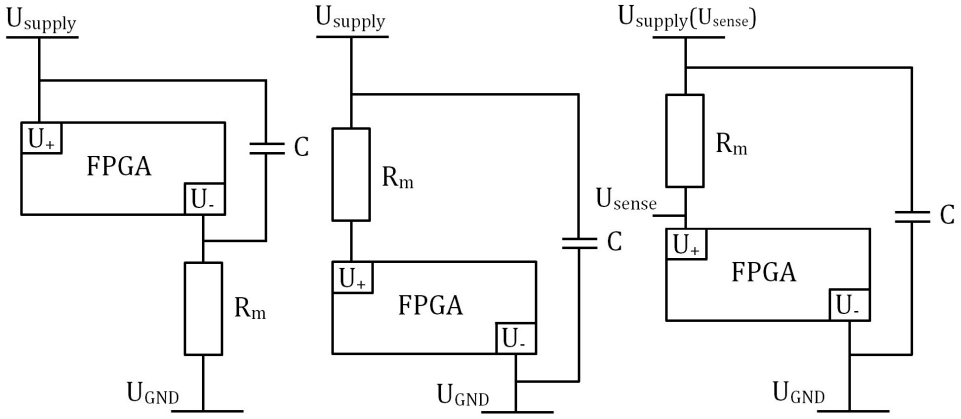

Abbildung 7.1: Konzepte verschiedener Messschaltungen

auf derselben oder einer unterschiedlichen Versorgungsspannungsebene, welche die Masseleitung mitbenutzen und somit zu einem verzerrten Ergebnis führen.

Das mittlere Teilbild in Abbildung 7.1 zeigt einen Messaufbau, der durch das Platzieren des Messwiderstands direkt vor dem Messobjekt die Stützkapazitäten umgeht. Um diesen Aufbau umzusetzen, müssen diese Kapazitäten zuerst auf dem Board entfernt werden. Dies führt dazu, dass es sich durch die Hardwareänderung nicht mehr um einen reinen Seitenkanalangriff handelt. Der Effekt, dass durch eine Stromänderung eine andere interne Spannungsversorgung U_+ anliegt, wird durch den zusätzlichen Spannungsabfall am Widerstand hervorgerufen. Diese Schwankung der Versorgungsspannung kann wegen ihrer geringen Rückwirkung auf die Verlustleistung ignoriert werden. Das rechte Teilbild bildet einen Messaufbau ab, der die Rückkopplung durch die Änderung des Spannungsabfalls am Messwiderstand verhindert. Das eingezeichnete Spannungsniveau U_{sense} fixiert die interne Versorgungsspannung auf ein bestimmtes Spannungsniveau, welches durch ein externes Netzgerät geregelt wird. Der externe Glättungskondensator C dient als rasche schwingungsarme Reaktion für die Versorgung des FPGAs und kann deswegen nicht eingespart werden. Jedoch wirkt dieser Bauteil gegen eine Änderung der Versorgungsspannung und verzögert die gesteuerte Versorgungsspannungsquelle.

7.2 Spartan-6 Entwicklungsboard

Für alle folgenden Untersuchungen wurde das „Atlys Spartan-6 FPGA Development Board“ von Digilent verwendet. Wie im Datenblatt [1] ersichtlich, verfügt das FPGA über einen 100 MHz CMOS Oszillator und 6.822 „Slices“. Jedes dieser „Slices“ beinhaltet vier „Look-up-table“ (LUT) mit jeweils sechs Eingängen, sowie acht Flip Flops. Zwei nebeneinander liegende „Slices“ ergeben zusammen einen „Configurable Logic Block“ (CLB).

Ein Viertel davon besteht nach [6] aus sogenannten „SLICEMs“ und verfügt über den vollen Funktionsumfang. Dazu gehört die Verwendung der LUT als 64 Bit verteiltes „Read Only Memory“ (RAM) oder als 32 Bit Schieberegister mit adressierbarer Länge. Der Ausgang eines LUTs kann dabei in ein Speicherelement des CLB führen. Mathematische Operationen können außerdem durch schnelle Übertragsketten optimiert werden. Weitere 25% setzen sich aus „SLICELs“ zusammen. Sie stellen eine abgespeckte Version eines „SLICEMs“ dar und verfügen deswegen über keine Schiebe- und RAM-Funktion. Die restliche Hälfte der verfügbaren „Slices“ bilden

die „SLICEXs“. Sie unterstützen eine ähnliche Struktur für die Programmierung wie „SLICELs“, mit dem Unterschied, dass sie keine arithmetische Übertragsoption anbieten.

Spannungsversorgung

Von einem externen 5 V Netzteil wird in der unveränderten Version des Development Boards in fünf unterschiedliche Spannungslevels transformiert:

Tabelle 7.1: Spartan 6 Development Board - Spannungsversorgung [1]

Name	Spannungsniveau	Zweck
V_{CC3V3}	$3,3$ V	FPGA I/O, Video, USB, Clock, ROM, Audio
V_{CC2V5}	$2,5$ V	FPGA Aux, VHDC, Ethernet PHY I/O, GPIO
V_{CC1V8}	$1,8$ V	DDR & FPGA DDR I/O
V_{CC1V2}	$1,2$ V	FPGA Core, Ethernet PHY Core
V_{TT}	$0,9$ V	DDR Termination Voltage

Für Seitenkanalanalysen spielt besonders das Spannungslevel V_{CC1V2} eine große Rolle, da dieses für die Versorgung der FPGA-Logik zuständig ist. Das Hilfsprogramm „Digilent Adept System V2.15.3“ greift auf mehrere 16 Bit „Delta-Sigma“ Analog-Digital-Wandler auf dem Entwicklungsboard zu und gibt somit einen guten ersten Einblick in die Zusammensetzung der Gesamtverlustleistung. Einen Screenshot davon zeigt Abbildung B.2 im Anhang.

Das Programm verrät den ungefähren Stromverbrauch in der Spannungsebene des „FPGA Cores“, der im Bereich zwischen 100 und 200 mA liegt. Im Zuge des Ladens einer Binärdatei in das FPGA werden jedoch in etwa 30 mA verbraucht. Um einen passenden Messwiderstand für einen stabilen Aufbau zu wählen, ist die Verlustleistung im Betrieb und während der Programmierung zu berücksichtigen. Der Spannungsabfall am Messwiderstand sollte daher so gering sein, dass diese Spannungsvariation weiterhin die minimalen und maximalen Versorgungsgrenzen einhält, und im Gegensatz dazu genügend Ausschlag aufweisen, damit eine verwertbare Messung aufgenommen werden kann.

Stützkapazitäten

Wie im Kapitel 7.1 erläutert, können Stützkapazitäten die Variation des Stromverbrauchs eines Gerätes nach außen hin verschleiern. An der Versorgungsleitung V_{CC1V2} liegen 43 dieser Kondensatoren für verschiedene Frequenzbereiche in Form von SMD-Bauteilen, die an der verwendeten Schaltung für die Untersuchung ausgelötet wurden. Die beiden Grafiken C.3 und C.4 im Anhang markieren in der Farbe Rot diese entfernten Kondensatoren.

Messschaltung

Ein $1,5\ \Omega$ Messwiderstand ist auf freien Lötstellen befestigt, um eine zusätzliche Induktivität in Form von Leitungen zu vermeiden. Diese Lötstellen sind durch Auslöten von Stützkondensatoren entstanden. Zwei zusätzliche parallel geschaltene Kapazitäten ($4,7$ µF und 47 nF), die nun außerhalb der Messung liegen, versuchen dem Aufbau wieder eine stabilere Versorgung zur Verfügung zu stellen (Mittleres Teilbild zu sehen in Abbildung 7.1).

Eine weitere Maßnahme, um die Stabilität der Versorgung der FPGA-Logik zu erhöhen, wird mit dem Einsatz eines externen, linearen Labor-Netzgeräts mit der Bezeichnung „Agilent E3631A“ umgesetzt. Einen vereinfachten Überblick über den gesamten Messaufbau, mit allen verwendeten Geräten und eingesetzter Software, sowie ihre Zusammenhänge zeigt Abbildung 7.2. Das Programmpaket MATLAB®, abgeleitet von „MATrix LABoratory“, übernimmt die Ansteuerung und die Verarbeitung der Daten. Es eignet sich besonders für Visualisierungen und statistische numerische Berechnungen, weswegen dieses Tool einen Seitenkanalangriff hervorragend unterstützen kann.

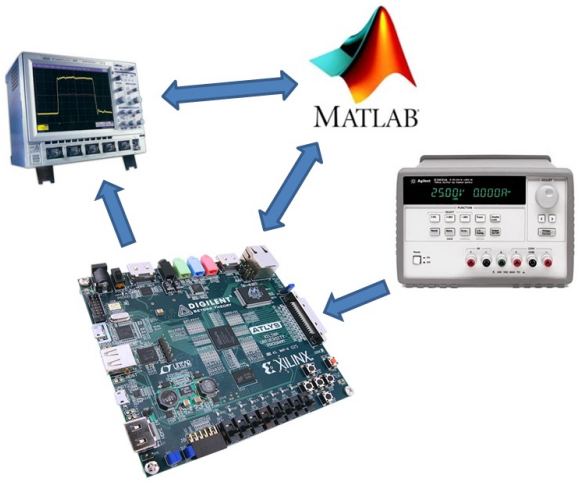

Abbildung 7.2: Übersicht Messaufbau

Bei dem verwendeten Oszilloskop handelt es sich um den „Waverunner LT584“ vom Hersteller „LeCroy“, welcher über seinen GPIB („General Purpose Interface Bus“) mit MATLAB® verbunden ist. Dieses digitale Speicher-Oszilloskop ermöglicht das Messen bis zu einer Bandbreite von 1 GHz und tastet bis zu 4 GS/s ab. Über zwei passive Tastköpfe vom selben Hersteller (Bezeichnung PP010) nimmt das Messgerät seine Daten auf, wobei eine Bandbreitenbegrenzung von 200 MHz das gemessene Signal am Oszilloskop filtert.

Die Aufgabe des ersten Tastkopfs besteht darin, sich mit dem gestarteten Verschlüsselungsalgorithmus zu synchronisieren. Dazu wird ein I/O-Pin am Board abgegriffen, der üblicherweise in einem realen Seitenkanalangriff nicht zur Verfügung steht. Da jedoch die meisten Verschlüsselungsvorgänge ein relativ dominantes Leistungsprofil aufweisen, wäre diese Synchronisierung auch ohne dieses Triggersignal möglich. Abbildung 7.3 zeigt diesen klaren Unterschied, wobei der linke Teil der Messung den Spannungseinbruch am FPGA vor dem Vorgang der Verschlüsselung darstellt. Der zweite Tastkopf misst den Spannungseinbruch an der FPGA-Versorgung, der durch den kurzzeitig erhöhten Stromverbrauch und den zusätzlichen Spannungsanstieg am Messwiderstand entsteht. Die externe Ansteuerung über MATLAB® sowie die Vorgabe der nötigen Verschlüsselungsparameter des jeweiligen Algorithmus an das FPGA findet über einen USB („Universal Serial Bus“)-UART („Universal Asynchronous Receiver Transmitter“) Port statt.

7.3 Erkenntnisse aus Voruntersuchungen

Im Vorfeld wurde bereits in verschiedenen Untersuchungen getestet, in welchem Ausmaß die Verlustleistung als Seitenkanal messbar ist.

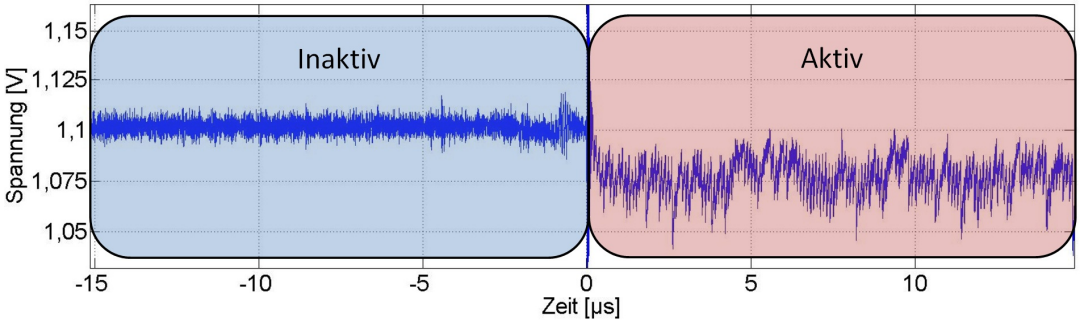

Abbildung 7.3: Spannungseinbruch vor und während einer Verschlüsselung

Wie bereits im vorherigen Abschnitt erwähnt, versorgt ein externes Netzgerät die FPGA-Logik. Außerdem helfen zwei Kondensatoren nahe an den Versorgungspins eine stabile Spannung bereitzustellen. Um nun den Stromverbrauch mit zwei „Single-Ended"-Tastköpfen zu messen, muss über die Spannungsdifferenz am Messwiderstand nach dem „Ohmschen Gesetz" durch den Widerstand dividiert werden.

Die Messaufnahme an zwei Messpunkten verursacht eine zusätzliche Abweichung, die durch den zweimal auftretenden Quantisierungsfehler und die zweimal auftretende Messungenauigkeit der Messschaltung entsteht. Die geringe Dynamik der stabilen Versorgungsspannung U_{supply} spielt für die Seitenkanalanalyse kaum eine Rolle und wurde in weiterer Folge als konstant angenommen. Dies führt dazu, dass nur der Spannungseinbruch am FPGA aufgenommen werden muss, der sich negativ proportional zur Stromänderung verhält. Eine vereinfachte Messschaltung bildet die Grafik 7.4 ab. In Kombination mit den Formeln 7.1 bis 7.3 lässt sich daraus die negative Proportionalität ableiten.

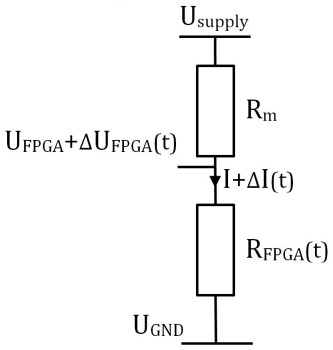

Abbildung 7.4: Negative Proportionalität zwischen Spannungseinbruch und Stromflussänderung

$$I = \frac{U_{supply} - U_{FPGA}}{R_m} \tag{7.1}$$

$$I + \Delta I(t) = \frac{U_{supply} - (\Delta U_{FPGA}(t) + U_{FPGA})}{R_m} \tag{7.2}$$

$$\Delta I(t) = -\frac{\Delta U_{FPGA}(t)}{R_m} \tag{7.3}$$

Um die Messbarkeit des Seitenkanals zu überprüfen und die Einstellungen an der Messschaltung zu optimieren, wurden Testprogramme auf die Entwicklungsplatine geladen und verglichen. Das Synthesetool „ISE Design Suite 14.7" trimmt nutzlosen Code weg. Eine externe Vorgabe und Ausgabe der Daten über den UART-Port verhindert diesen unerwünschten Effekt.

Ein Testprogramm, welches bis zu 8.000 Register gleichzeitig toggelt, soll die Linearität von Registeränderungen zeigen. Im ersten Taktschritt negieren für diesen Zweck alle Register und in den folgenden zwei Schaltzeitpunkten jeweils nur eine Hälfte der Speicherzellen. Aus den Messauf-

zeichnungen, die in der Abbildung 7.5 sichtbar sind, können somit zwei Erkenntnisse gewonnen werden, die im finalen Aufbau beobachtet werden können.

- Der Spannungseinbruchsunterschied zwischen dem gleichzeitigen Wechsel von der Hälfte der Register zu allen Registern verhält sich nahezu linear.
- Die Differenz zwischen dem Umschalten aller Zellen vom ungesetzten zum gesetzten Zustand und dem umgekehrten Szenario ist nicht merkbar.

Bei einer Periodendauer von 100 ns sieht man in der Grafik den Spannungseinbruch am FPGA zum Schaltzeitpunkt A, in dem 8.000 Bit gleichzeitig ihren Zustand wechseln. Hingegen zeigt der Zeitpunkt B die Negierung von 4.000 Speicherzellen.

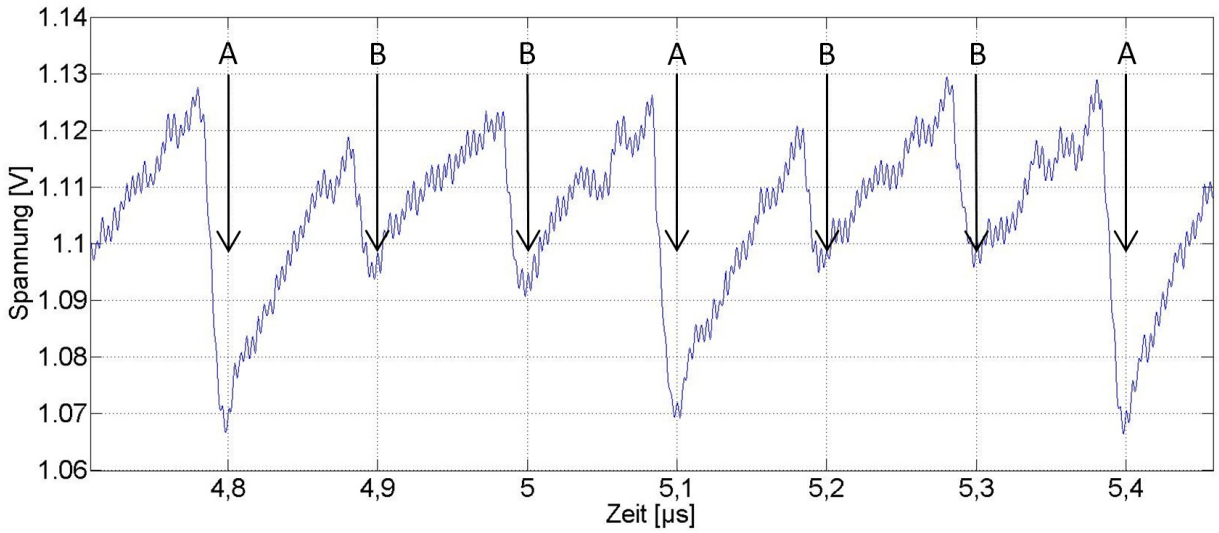

Abbildung 7.5: Messung von Registerzustandswechsel in einem FPGA

Es hat sich auch gezeigt, dass ein Signalwechsel am herausgeführten Ausgangspin am „Atlys“ FPGA-Board kurzzeitig zu starken Schwingungen führt. Ein solcher Signalwechsel wird als Trigger für die Messung in dieser Arbeit verwendet und verhindert somit eine eindeutige Messaufnahme der Verlustleistung zum Triggerzeitpunkt. Durch die Wahl eines früheren Triggerimpuls für die Synchronisation mit dem Oszilloskop, ist es möglich diese unerwünschten Effekte in der Testumgebung zu umgehen. Abbildung 7.6 stellt dieses schwingende Signal, verursacht durch die logische Änderung eines Ausgangspins, dar. Die Störung beeinflusst in dieser Messaufnahme zwei Taktzyklen bei einer Taktfrequenz von 10 MHz.

Mit der Anwendung einer FFT („Fast Fourier Transform“) auf das gemessene Seitenkanalsignal kann ein erster Einblick auf das Leistungsspektrum während eines Verschlüsselungsvorgangs geschaffen werden. Es hat sich gezeigt, dass eine geringere Taktfrequenz am FPGA-Entwicklungsboard weniger anfällig auf hochfrequente Störungen ist und somit ein saubereres Signal für die Messung verursacht. Das aufgenommene Spektrum bei einer Taktfrequenz von 10 MHz zeigt Abbildung 7.7. Auf der logarithmischen X-Achse kann man einerseits Oberwellen an multiplikativen Vielfachen der verwendeten Taktfrequenz erkennen und andererseits den farblich markierten Frequenzbereich, der für die Messung der Seitenkanalinformationen ausschlaggebend ist.

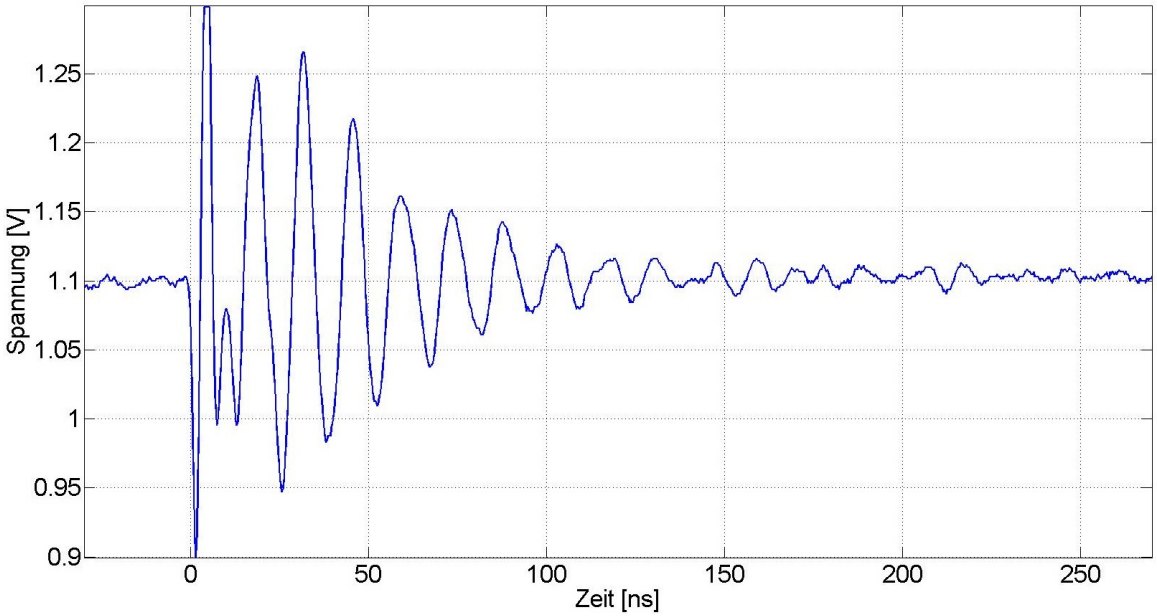

Abbildung 7.6: Störsignal verursacht durch Trigger

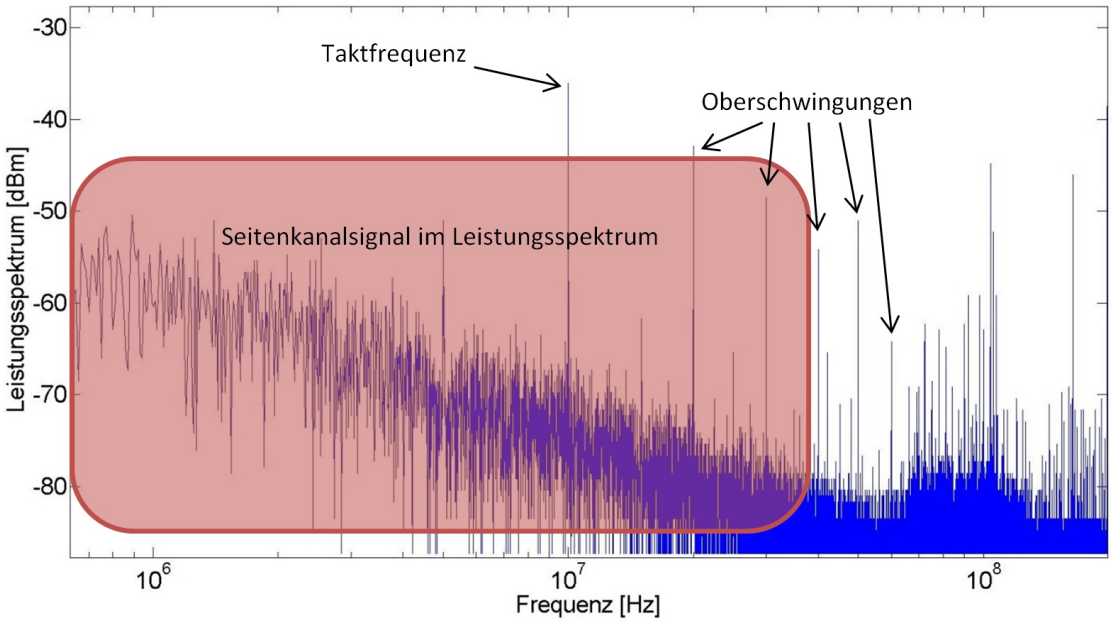

Abbildung 7.7: Leistungsspektrum eines Verschlüsselungsvorgangs

8 Zusammenfassung

Diese Zusammenfassung soll einen abschließenden groben Überblick über diese Arbeit schaffen und die wesentlichen Argumentationspunkte herausstreichen. Des Weiteren werden gewonnene Erfahrungen und Erkenntnisse präsentiert und die Ergebnisse mit der Aufgabenstellung verglichen. Im letzten Unterabschnitt 8.2 werden Ideen für weiterführende Forschungsarbeiten vorgestellt, denen diese Arbeit als Basis dienen könnte.

8.1 Ergebnisse und Erkenntnisse

Die Detektion von Algorithmen in einem FPGA stellt sich oft als notwendige Vorbedingung für einen möglichen Seitenkanalangriff dar. Ein Angreifer kann durch die Kenntnis des Verschlüsselungsalgorithmus seinen Angriff auf die untersuchte Implementierung verfeinern und die Seitenkanalinformationen richtig interpretieren.

Es hat sich gezeigt, dass ein gut eingestellter Messaufbau für die Untersuchung der Seitenkanalinformationen und somit eines Erfolgs ausschlaggebend ist. Neben der richtigen Wahl des Messwiderstands, sodass weiterhin die Betriebsgrenzen eingehalten werden können, führen zusätzliche Leitungsinduktivitäten im Messaufbau zu einer starken Beeinflussung der gemessenen Signale. Der durch die längere Messleitung entstehende Schwingkreis kann durch eine lokale, möglichst nahe am FPGA befindliche, Messschaltung geschwächt werden. Direkt am untersuchten Device anliegende Stützkapazitäten können die Seitenkanaleffekte in der Ausführung kompensieren und sollten deswegen entfernt oder außerhalb der Messung verlegt werden.

Eine andere Quelle für starke Schwingungen im Messsignal wird durch eine hohe Taktfrequenz im Zielsystem verursacht. Eine vorgenommene Frequenzreduktion lässt deshalb ein saubereres messbares Signal entstehen und die Extraktion detaillierter Informationen zu.

Bei der Umsetzung von verschiedenen Algorithmen stellte sich heraus, dass sich nicht alle verfügbaren mathematischen Standardoperationen für die Verschlüsselung eignen. Daher kommen auf einem FPGA nur Algorithmen in Frage, die auf ressourcenschonende Operatoren zurückgreifen. Die Blakley- und die Montgomery-Multiplikation zählen zu diesen Algorithmen und können mit Schiebe-, Vergleichs-, Additions- und Subtraktionsoperationen eine modulare Multiplikation in mehreren Taktschritten umsetzen.

Verschiedene Seitenkanalangriffsvektoren machen es möglich, den privaten Schlüssel aus jedem der vier untersuchten Algorithmen in einer FPGA Implementierung zu rekonstruieren. Tabelle 8.1 zeigt, welcher Algorithmus mit welcher Vorgehensweise erfolgreich attackiert wurde.

Tabelle 8.1: Erfolgreich durchgeführte Seitenkanalangriffe auf Verschlüsselungsimplementierungen

Algorithmus	Angriffsmethoden
„Quadrieren und Multiplizieren“-„Blakley“-Algorithmus	-SPA mit kleinem Extremwert -SPA mit N-1 Extremwert -Correlation Attack
„Chinesischer Restsatz“-„Blakley“-Algorithmus	-Rechenzeitangriff
„Quadrieren und Multiplizieren“-„Montgomery“-Algorithmus	-SPA mit kleinem Extremwert im Montgomery-Raum -SPA mit N-1 Extremwert -Correlation Attack
„Chinesischer Restsatz“-„Montgomery“-Algorithmus	-Rechenzeitangriff

Die Analyse der Verlustleistung eines „Field Programmable Gate Arrays“ während der Abarbeitung von Extremwerten zeichnet sich als besonders robuster Angriffsvektor aus. Solange ein Unterschied zwischen den, durch den Extremwert bestimmten Operationen erkennbar ist, kann der Schlüssel implementierungsunabhängig extrahiert werden. Je nach Algorithmus schlagen die vorgegebenen Extremwerte unterschiedlich an. Deswegen eignet sich dieser Angriffsvektor besonders als Detektionsmethode für eine unbekannte Implementierung.

Im Gegensatz zu einer Blakley-Multiplikation benötigt eine Multiplikation im Montgomery-Raum durch den einfacheren Modulo-Reduziervorgang weniger Logik in einem FPGA Design. Diese Vereinfachung reduziert die Dynamik im Seitenkanalsignal und dadurch die Lesbarkeit für eine SPA. Die Transformation in den Montgomery-Raum wirkt sich auf die Extremwertvorgabe aus und schafft somit ein starkes Unterscheidungsmerkmal.

Korrelationsangriffe benutzen für den Abgleich ein mathematisches Simulationsmodell zur Verlustleistungsabschätzung. Dieses Modell wendet in dieser Diplomarbeit die Kombination aus der Hamming-Distanz und zusätzlich die kurzzeitige logische Falschstellung (Glitches), hervorgerufen durch verschiedene Laufzeiteffekte in zusammenhängenden asynchronen Schaltteilen, an. Der letztgenannte Effekt wirkt sich in den untersuchten Implementierungen, die mehrere voneinander abhängige Bedingungsprüfungen hintereinander in einem Takt besitzen, außerordentlich dominant aus.

Ein RSA Verschlüsselungsalgorithmus, der auf dem chinesischen Restsatz Theorem beruht (CRT-RSA), verwendet die beiden Schlüsselerzeugerprimzahlen für die Berechnung. Dadurch kann man den Rechenvorgang in zwei kleinere modulare Potenzierfunktionen zerlegen und infolgedessen Rechenzeit, sowie Hardwareressourcen sparen. Eine nötige Reduzierung der Eingangsdaten mit einer Modulo-Operation steht immer am Beginn dieses Rechenalgorithmus und kann im Zuge seiner Ausführungsdauer den geheimen Schlüssel verraten. Verschiedene Implementierungen einer Modulo-Einheit wurden in dieser Diplomarbeit erfolgreich attackiert. Über unterschiedliche Ausführungszeiten, die von der Größe der anliegenden Eingangsdaten abhängen, kann damit CRT-RSA in einem unbekannten System entlarvt werden.

Spezielle Rechenvorschriften, wie die Montgomery-Multiplikation und das modulare Potenzieren auf Basis des chinesischen Restsatzes, haben aufgrund ihrer Struktur verschiedene mathemati-

sche Operationen zu erfüllen. Vergleicht man den Unterschied zwischen diesen, erkennt man oft deutliche Unterschiede im Verlustleistungsmuster in der Ausführung, die für die Identifikation einsetzbar sind. Dafür kann auch die Variation der gemessenen Leistung genutzt werden. Das funktioniert besonders dann, wenn ein Zusammenhang mit der Änderung von bestimmten Abschnitten und beobachtbarer veränderter Verschlüsselungsparameter besteht.

Ein Angriff über die Korrelation mit einem Simulationsmodell benötigt eine Sammlung von mehreren möglichen Modellen, um diese aus einer Blackbox demaskieren zu können. Auch wenn das Modell aus dem VHDL Code und nicht aus einer detaillierteren Abstraktionsebene generiert wird, bleiben dem Entwickler viele Freiheiten in der Implementierung, wodurch diese Detektionsmethode für einen FPGA nur beschränkt Einsatz findet.

Für die Detektion von Algorithmen eignen sich nicht alle Seitenkanalangriffsvektoren: Template-Angriffe und „High-Order Differential Power Analysis“ benötigen Detailinformationen oder eine möglichst exakte Kopie der Orginalimplementierung, um auf den Algorithmus in einer Blackbox rückschließen zu können.

Die zwei wichtigsten Kategorien von Gegenmaßnahmen, die die Identifizierung mit den vorgestellten Techniken erschweren oder verhindern, nennen sich Verstecken und Maskieren. Einerseits verhindert das Maskieren der Eingangsdaten durch das Randomisieren, dass Extremwerte in der Ausführung Informationen preisgeben und vereiteln deswegen auch die Erkennung des Algorithmus. Auf der anderen Seite kann das Verstecken eine genauere Analyse mit Hilfe der Rechenzeit und der untersuchten Abschnitte verkomplizieren.

Aus dieser kurzen Zusammenfassung der Ergebnisse und Erkenntnisse lässt sich feststellen, dass verschiedene Methoden zur Detektion verwendet und auch die untersuchten Algorithmen in ihrer Implementierung identifiziert werden können. Erkennungsverfahren mit einem generischen Konzept stellen sich im Zuge dieser Arbeit als besonders effektiv heraus. Hingegen eignen sich von Detailinformationen abhängige Seitenkanalangriffskonzepte nicht.

SPA mit Extremwerten und der Rechenzeitangriff setzen als Bedingung voraus, dass der Angreifer Daten in der Entschlüsselungseinheit vorgeben kann. Im Vergleich dazu, ist für das Finden von Teilabschnitten in der Verlustleistung mit unterschiedlichen Mustern und die Abhängigkeiten der Änderungen dieser Abschnitte keine Vorgabe nötig. Anhand von Extremwerten lässt sich am besten ein Algorithmus in einer Blackbox erfassen, da sich die Unterscheidung durch die starke Ausprägung der Effekte besonders einfach nachvollziehen lässt.

Der Schutzmechanismus, der die chiffrierte Nachricht maskiert, ist leichter umsetzbar als alternative Maßnahmen, die auf das Verstecken abzielen und besonders effektiv gegen einen Angriff mit SPA mit speziellen Nachrichten. Als die sicherste Implementierung hat sich der „Chinesischer Restsatz“-„Montgomery“-Algorithmus herausgestellt, da dieser im Vergleich zu den anderen die geringsten Hardwareressourcen benötigt und damit auch das kleinste SNR im Seitenkanal produziert.

Als Fazit kann außerdem festgehalten werden, dass es sich in der Praxis lohnt, verschiedene Methoden zu testen und selbst zu entscheiden, welche im vorliegenden Fall die aufschlussreichsten Informationen preisgeben.

8.2 Ausblick

Das Forschungsgebiet, welches sich mit der Sichtbarkeit von Blackbox-Systemen befasst, ist wenig untersucht und lässt viele weiterführende Arbeiten zu. Diese Diplomarbeit befasst sich mit vier für die Praxis relevanten Algorithmen, die auf eine mögliche Unterscheidung analysiert wurden. In der Anwendung befinden sich aber noch andere Algorithmen, wie die „m-ary"- oder die „Sliding Window"-Rechenvorschrift, die in dieser Arbeit nicht behandelt wurden und noch untersucht werden könnten. Verschiedene Fenstergrößen, die eine zusätzliche Variation in der Implementierung verursachen, könnten dabei die Detektion verkomplizieren.

Als Vorbereitung für einen Angriff kann die Erkennung von Gegenmaßnahmen in einem unbekannten System ebenfalls wert sein, sich damit tiefgehender zu beschäftigen. Erst wenn der Schutzmechanismus bekannt ist, kann der Angreifer versuchen diesen mit der entsprechenden Seitenkanalinterpretierung zu umgehen.

Neben der RSA Verschlüsselungsmethode könnten auch andere Methoden, wie „Elliptic Curve Cryptography", auf Implementierungsdetails untersucht werden. Andere Verschlüsselungen weisen divergente Eigenschaften auf. Dies würde zu einer notwendigen Anpassung der Detektionsmethoden und neuer Interpretationsmöglichkeiten der Informationen führen.

Die vorliegende Arbeit beschäftigt sich verstärkt mit dem Seitenkanal „Verlustleistung". Für eine Analyse wäre zum Beispiel auch die elektromagnetische Abstrahlung nutzbar. Die Anwendung der hier vorgestellten Konzepte von Erkennungsmerkmalen könnte dadurch auch in diesem Kanal geprüft und erweitert werden.

Mit einer zusätzlichen Verifikation der Detektionsergebnisse könnte aus verschiedenen praktischen Implementierungen der Verschlüsselungen eine Erfolgsquote angegeben und damit die Effektivität besser bewertet werden. Wobei sich einige Konzepte auch auf nicht hardwareprogrammierbare Systeme übertragen lassen.

Schafft man es, diese soeben beschriebenen Forschungslücken hinreichend zu erarbeiten und damit einige der offenen Punkte zu klären, kann dazu ein automatisiertes Detektionstool entwickelt werden.

Anhang

Im Anhang befinden sich Informationen zur Kryptographie auf elliptischen Kurven, auf die in der Arbeit öfters Bezug genommen wird, sowie Auszüge aus dem „Atlys“ Datenblatt und eine FPGA-Stromverbrauchsmessung mit dem „Digilent Adept System“-Tool.

A.1 Kryptographie auf elliptischen Kurven

Das Verschlüsselungsverfahren ECC („Elliptic Curve Cryptography“) zählt zu den asymmetrischen Verschlüsselungsmethoden und verwendet nach [Bro09] geometrische Objekte (elliptische Kurven) und Operationen darauf. Mit diesen Operationen stellt die elliptische Kurve einen endlichen Körper da, der für Verschlüsselungen verwendet werden kann. In der Praxis werden dazu zwei Typen von endlichen Körpern eingesetzt [Bro09, S. 6-7]:

- Die Formel für eine allgemeine kryptographische Kurve im „Prime“-Körper $\mathbb{F}_p$, wobei p für die Anzahl der Elemente im Körper steht und eine Primzahl repräsentiert, lautet

$$y^2 \equiv x^3 + a \cdot x + b \mod p, \text{ mit } x, y, a, b \in \mathbb{F}_p \tag{A.1}$$

$$4 \cdot a^3 + 27 \cdot b^2 \not\equiv 0 \mod p \tag{A.2}$$

- Der binäre Körper im $\mathbb{F}_{2^m}$ mit $m \geq 1$ und $b \neq 0$ beinhaltet 2^m Elemente und seine mathematische Repräsentation lautet

$$y^2 + x \cdot y = x^3 + a \cdot x^2 + b \text{ im } \mathbb{F}_{2^m}, \text{ mit } x, y, a, b \in \mathbb{F}_{2^m} \tag{A.3}$$

Auf einer elliptischen Kurve wird im Allgemeinen zwischen zwei Operationen unterschieden:

- Addition - zwei verschiedene Punkte werden addiert.
- Verdoppelung - ein Punkt wird zu sich selbst addiert.

Diese beiden Operationen werden in der Arbeit [Swe08, S. 52-55] ausführlich vorgestellt. Während die Addition so definiert ist, dass eine Gerade durch die zwei zu addierenden Punkte gelegt wird und der daraus ergebende zusätzliche Schnittpunkt gespiegelt an der x-Achse das Ergebnis liefert, ergibt sich hingegen der Schnittpunkt in der Verdoppelungsoperation durch die Tangente

im aktuellen Punkt. Die Abbildung Figure A.1 zeigt die vereinfachte Addition der zwei Punkte P und Q über die reellen Zahlen. Das neutrale Element $\mathcal{O}$ liegt nach in der Unendlichkeit. Dies bietet die Möglichkeit allgemein einen inversen Punkt $-(x, y) = (x, -y)$ einzuführen, sodass $(x, y) + (x, -y) = \mathcal{O}$ gilt.

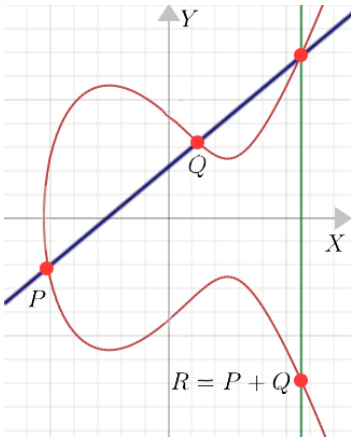

Abbildung A.1: Punktaddition in der Kurve $y^2 = x^3 - 3 \cdot x + 3$ über die reellen Zahlen [Swe08, S. 54]

Der Schnittpunkt berechnet sich laut [Bro09, S. 6-7] durch das Gleichsetzen der Kurve mit der Geraden und führt durch die Negierung der Y-Achse zum gesuchten Ergebnis. Für die oben erwähnte Kurve im $\mathbb{F}_p$ errechnen sich die zwei Punkte P_1 und P_2 zu

$$P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2) \tag{A.4}$$

$$x_3 \equiv s^2 - x_1 - x_2 \mod p \tag{A.5}$$

$$y_3 \equiv s \cdot (x_1 - x_3) - y_1 \mod p \tag{A.6}$$

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \mod p, \quad \text{falls } P_1 \neq P_2 \text{ (Addition)} \\ \frac{3 \cdot x_1^2 + a}{2 \cdot y_1} & \mod p, \quad \text{sonst (Verdopplung).} \end{cases} \tag{A.7}$$

In [Bro09, S. 7-8] wird ebenso der Unterschied in den Operationen auf die elliptischen Kurven im Körper $\mathbb{F}_{2^m}$ beschrieben. Dieser äußert sich durch das inverse Element $-(x, y) = (x, x + y)$, das Reduktionspolynom $p(x)$ anstatt der Modulo-Operation und die resultierende Lösung des dritten Schnittpunkts. Das Ergebnis der Punkt-Addition zweier verschiedener Punkte P_1 und P_2 lautet deshalb

$$x_3 = s^2 + s + x_1 + x_2 + a \text{ im } \mathbb{F}_{2^m} \tag{A.8}$$

$$y_3 = s \cdot (x_1 + x_3) + x_3 + y_1 \text{ im } \mathbb{F}_{2^m} \ . \tag{A.9}$$

Für die Punkt-Verdoppelung gilt hingegen die Formel

$$x_3 = s^2 + s + a \text{ im } \mathbb{F}_{2^m} \tag{A.10}$$

$$y_3 = x_1^2 + (s + 1) \cdot x_3 \text{ im } \mathbb{F}_{2^m} \ , \tag{A.11}$$

wobei gilt

$$s = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1} \text{ im } \mathbb{F}_{2^m}, & \text{falls } P_1 \neq P_2 \text{ (Addition)} \\ x_1 + \frac{y_1}{x_1} \text{ im } \mathbb{F}_{2^m}, & \text{sonst (Verdopplung).} \end{cases} \tag{A.12}$$

A.1.1 Wahl der Kurvenparameter und Patente

„Kryptosysteme basierend auf elliptischen Kurven sind prinzipiell patentfrei. Es gibt allerdings einige Algorithmen und Techniken zur effizienten Implementierung, auf die Patentansprüche erhoben werden“ [Osw11, S. 27]. Ein internationales Konsortium mit dem Namen „Standards for Efficient Cryptography Group“ versucht eine Plattform für Interoperabilität zu schaffen und stellt eine Liste mit empfohlenen freien Parametern [Bro10] für viele verschiedene elliptische Kurven zur Verfügung.

Kurvenparameter in $\mathbb{F}_p$ und $\mathbb{F}_{2^m}$

Für die Beschreibung der Parameter einer elliptischen Kurve in $\mathbb{F}_p$ muss ein Sechstupel an Informationen fixiert sein [Bro09, S. 15]:

$$T = (p, a, b, G, n, h) \tag{A.13}$$

1. Die Zahl p spezifiziert den endlichen Körper $\mathbb{F}_p$.
2. Die Werte a und b nennen die Unbekannten in der Kurvengleichung.
3. Der Basis Punkt G liegt auf der Kurve und wird für die skalare Multiplikation verwendet.
4. Die Primzahl n kennzeichnet die Ordnung des Punktes G und somit die kleinste nicht negative Zahl, welche die Gleichung $n \cdot G = \mathcal{O}$ erfüllt.
5. Der Parameter h steht für den Cofaktor ($h = \frac{\#E(\mathbb{F}_p)}{n}$)

Die Beschreibung in $\mathbb{F}_{2^m}$ erfolgt ähnlich zu den Parametern in $\mathbb{F}_p$ durch ein Septupel [Bro09, S. 16]:

$$T = (m, f(x), a, b, G, n, h) \tag{A.14}$$

Der entscheidende Unterschied liegt im unreduzierbaren Polynom $f(x)$ des Grades m.

A.1.2 Skalare Punkt-Multiplikation

Das Wandern auf der elliptischen Kurve, welches die Basis dieser Verschlüsselung darstellt, kann auch durch eine skalare Multiplikation $k \cdot P = \underbrace{P + P + P + P \ldots + P + P}_{\text{k mal}}$ angeschrieben werden. Dafür existieren Algorithmen, die eine Analogie mit dem bereits vorgestellten modularen Potenzieren in Kapitel 3.2.2 aufweisen.

Ein Beispiel dafür ist der Algorithmus „binäres modulares Potenzieren“ in Abschnitt 3.1. Er kann durch Ersetzen der Schritte Quadrieren und Multiplizieren mit den ECC-Operationen Verdoppelung und Addition, sowie die Verwendung des skalaren Werts k anstatt des Schlüssels e, so modifiziert werden, so dass dieser die gleiche Funktionalität bietet [HHM01].

A.1.3 Punkt Addition - Projektive Koordinaten

Nach [HVM04, S. 68-88] kann neben der Reihenfolge und der Vorgangsweise in der skalaren Punkt-Multiplikation auch die Grundoperation Punkt-Addition auf verschiedene Arten implementiert werden. Durch die Transformation in ein anderes Koordinatensystem, in dem für die Addition und Verdoppelung keine Inversion benötigt wird, ergeben sich erhebliche Performance-Verbesserungen. Die Koordinaten (X, Y, Z) im neuen Koordinatensystem korrespondieren zu den ursprünglichen Koordinaten $(X/Z, Y/Z)$ für Standard projektive Koordinaten und $(X/Z^2, Y/Z^3)$ für Jacobi projektive Koordinaten.

B.2 Digilent Adept System

Mit dem Tool „Digilent Adept System V2.15.3“ konnten erste Abschätzungen über die Verlustleistung der untersuchten FPGA-Plattform durchgeführt werden. Der Screenshot im Bild B.2 zeigt die verschiedenen Spannungsversorgungsebenen und ihren zugehörigen Stromverbrauch im Normalbetrieb.

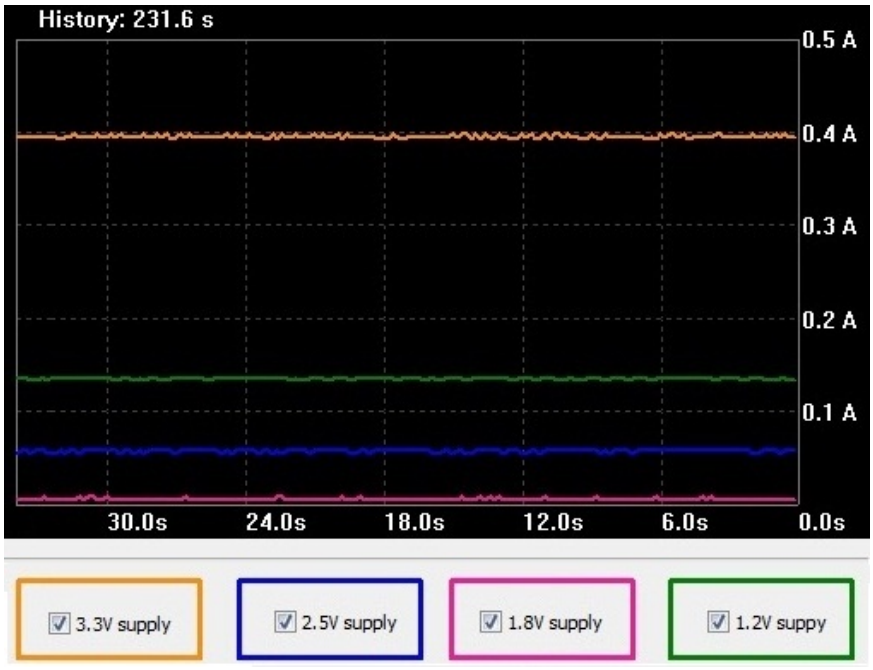

Abbildung B.2: Screenshot - „Digilent Adept System V2.15.3“

C.3 Entfernte Stützkapazitäten

Die Abbildungen C.3 und C.4 geben einen Überblick über Stützkapazitäten, die die Seitenkanalmessung an einem „Atlys“ Entwicklungsboard verhindern. Die Kondensatoren glätten den kurzzeitigen höheren Stromverbrauch und wurden deshalb ausgelötet.

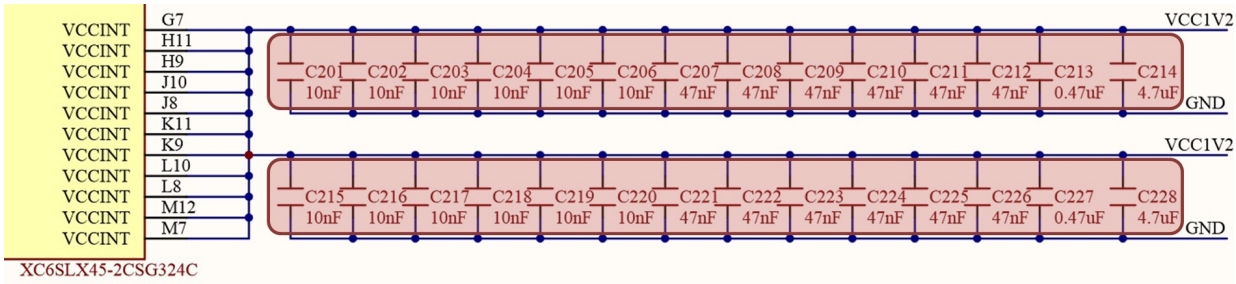

Abbildung C.3: Einige entfernte Stützkapazitäten am FPGA Versorgungseingang V_{CCINT}[2]

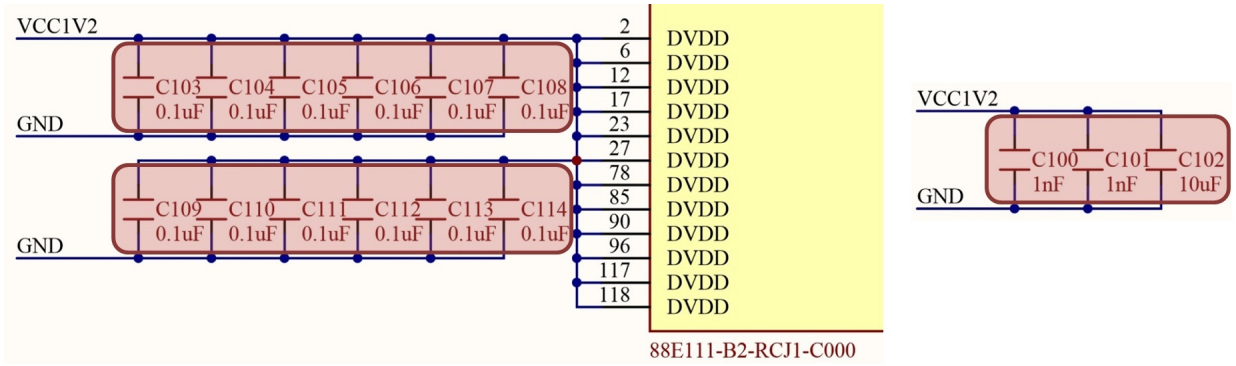

Abbildung C.4: Einige entfernte Stützkapazitäten nahe der Ethernetcontroller-Versorgung V_{DVDD}[2]

Wissenschaftliche Literatur

[AARR03] Agrawal, D. ; Archambeault, B. ; Rao, J. R. ; Rohatgi, P.: The EM side-channel(s). In: *Cryptographic Hardware and Embedded Systems-CHES 2002*, Springer, 2003. – ISBN 978–3540004097, S. 29–45

[AFV07] Amiel, F. ; Feix, B. ; Villegas, K.: Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In: *Selected Areas in Cryptography* Springer, 2007. – ISBN 978–3540773597, S. S. 110–125

[AL02] Adams, C. ; Lloyd, S.: Understanding PKI: Concepts, Standards, and Deployment Considerations, Sams, 2002. – ISBN 978–0672323911

[Bar12] Barron, J. W.: Thesis: RSA Power Analysis Obfuscation: A Dynamic FPGA Architecture, Air Force Institute of Technology, 2012

[Bau12] Bauer, S.: Attacking Exponent Blinding in RSA without CRT. In: *Constructive Side-Channel Analysis and Secure Design*, Springer, 2012. – ISBN 978–3642299117, S. 82–88

[BCO04] Brier, E. ; Clavier, C. ; Olivier, F.: Correlation Power Analysis with a Leakage Model. In: *Cryptographic Hardware and Embedded Systems-CHES 2004*, Springer, 2004. – ISBN 978–3540226666, S. 16–29

[BDL97] Boneh, D. ; DeMillo, R. A. ; Lipton, R. J.: On the Importance of Checking Cryptographic Protocols for Faults. In: *Advances in Cryptology—EUROCRYPT'97* Springer, 1997. – ISBN 978–3540629757, S. 37–51

[BK08] Backes, M. ; Köpf, B.: Formally Bounding the Side-Channel Leakage in Unknown-Message Attacks. In: *Computer Security-ESORICS 2008*, Springer, 2008. – ISBN 978–3540883128, S. 517–532

[Bla83] Blakley, G. R.: A Computer Algorithm for Calculating the Product AB Modulo M. In: *IEEE Transactions on Computers, Vol. C-32, No. 5*, Institute of Electrical and Electronics Engineers, 1983. – ISSN 0018–9340, S. 497–500

[BO08] Bayam, K. A. ; Ors, B.: Differential Power Analysis Resistant Hardware Implementation of The RSA Cryptosystem. In: *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium*, Institute of Electrical and Electronics Engineers, 2008. – ISBN 978–1424416837, S. 3314–3317

[Bro09] Brown, D. R. L.: Standards for Efficient Cryptography SEC 1: Elliptic Curve Cryptography - Version 2, Certicom Research, 2009

[Bro10] Brown, D. R. L.: Standards for Efficient Cryptography SEC 2: Recommended Elliptic Curve Domain Parameters - Version 2, Certicom Research, 2010

[BSI14] Kryptographische Verfahren: Empfehlungen und Schlüssellängen Bundesamt für Sicherheit in der Informationstechnik, 2014

[CRR03] Chari, S. ; Rao, J. R. ; Rohatgi, P.: Template Attacks. In: *Cryptographic Hardware and Embedded Systems-CHES 2002*, Springer, 2003. – ISBN 978–3540004097, S. 13–28

[DH76] Diffie, W. ; Hellman, M.: New Directions in Cryptography. In: *IEEE Transactions on Information Theory, Vol. IT-22, No. 6*, Institute of Electrical and Electronics Engineers, 1976, S. 644–654

[Die06] Dierks, T.: RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1, Internet Engineering Task Force, 2006

[Eck11] Eckert, Claudia: IT-Sicherhei: Konzepte - Verfahren - Protokolle, Oldenbourg Wissenschaftsverlag, 2011. – ISBN 978–3486706871

[ElG85] ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: *Advances in cryptology : proceedings of CRYPTO 84*, Springer, 1985. – ISBN 978–3540395683, S. 10–18

[EPW11] Eisenbarth, T. ; Paar, C. ; Weghenkel, B.: Building a Side Channel Based Disassembler. In: *Transactions on computational science X*, Springer, 2011. – ISBN 978–3642174988, S. 78–99

[Eva11] Evans, D.: CISCO White Paper - The Internet of Things: How the Next Evolution of the Internet is Changing Everything, 2011

[Fis10] Fischer, R.: Dissertation: Methodik für die Verlustleistungsabschätzung von Prozessoren mit Pipeline-Strukturen, Dr. Hut, 2010. – ISBN 978–3868534788

[GBPV10] Gierlichs, B. ; Batina, L. ; Preneel, B. ; Verbauwhede, I.: Revisiting higher-order DPA attacks. In: *Topics in Cryptology-CT-RSA 2010*, Springer, 2010. – ISBN 978–3642119248, S. 221–234

[GM11] Gueneysu, T. ; Moradi, A.: Generic Side-Channel Countermeasures for Reconfigurable Devices. In: *Cryptographic Hardware and Embedded Systems-CHES 2011*, Springer, 2011. – ISBN 978–3642239502, S. 33–48

[GM13] Gosheblagh, R. O. ; Mohammadi, K.: International Journal of Computer Applications: Dynamic partial based Single Event Upset (SEU) injection platform on FPGA, 2013. – ISSN 0975–8887

[Har06] Harris, B.: RFC 4432: RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol, Internet Engineering Task Force, 2006

[Hei10] Heiman, G. W.: Basic Statistics for the Behavioral Sciences, South Western Educ Pub, 2010. – ISBN 978–0840031433

[HHM01] Hankerson, D. ; Hernandez, J. L. ; Menezes, A.: Software Implementation of Elliptic Curve Cryptography over Binary Fields. In: *Cryptographic Hardware and Embedded Systems-CHES 2000* Springer, 2001. – ISBN 978–3540414551, S. 1–24

[HMH+13] Heyszl, J. ; Merli, D. ; Heinz, B. ; Santis, F. D. ; Sigl, G.: Strengths and Limitations of High-Resolution Electromagnetic Field Measurements for Side-Channel Analysis. In: *Smart Card Research and Advanced Applications*, Springer, 2013. – ISBN 978–3642372872, S. 248–262

[HPS08] Hoffstein, J. ; Pipher, J. ; Silverman, J. H.: An Introduction to Mathematical Cryptography, Springer, 2008. – ISBN 978–0387779942

[HVM04] Hankerson, D. ; Vanstone, S. ; Menezes, A.: Guide to Elliptic Curve Cryptography, Springer, 2004. – ISBN 978–0387952734

[JY03] Joye, M. ; Yen, S.-M.: The Montgomery Powering Ladder. In: *Cryptographic Hardware and Embedded Systems-CHES 2002*, Springer, 2003. – ISBN 978–3540004097, S. 291–302

[KJJ99] Kocher, P. C. ; Jaffe, J. ; Jun, B.: Differential Power Analysis. In: *Advances in*

Cryptology—CRYPTO'99 Springer, 1999. – ISBN 978–3540663478, S. 388–397

[KJJR11] Kocher, P. ; Jaffe, J. ; Jun, B. ; Rohatgi, P.: Introduction to Differential Power Analysis. In: *Journal of Cryptographic Engineering, Vol. 1, No. 1*, Springer, 2011. – ISSN 2190–8508, S. 5–27

[KK94] Ç. K. Koç: High Speed RSA Implementation, RSA Laboratories, 1994

[KK10] Karpfinger, C. ; Kiechle, H.: Kryptologie: Algebraische Methoden und Algorithmen Vieweg+Teubner Verlag, 2010. – ISBN 978–3834808844

[Kob94] Koblitz, Neal: A Course in Number Theory and Cryptography, Springer, 1994. – ISBN 978–0387942933

[Koc96] Kocher, P. C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: *Advances in Cryptology—CRYPTO'96* Springer, 1996. – ISBN 978–3540615125, S. 104–113

[Kre98] Kredler, C.: Einführung in die Wahrscheinlichkeitsrechnung und Statistik, 1998

[KSRHP07] Koç, Ç. K. ; Saqib, N. A. ; Rodríguez-Henríquez, F. ; Pérez, A. D.: Cryptographic Algorithms on Reconfigurable Hardware, Springer, 2007. – ISBN 978–0387366821

[KSWH98] Kelsey, J. ; Schneier, B. ; Wagner, D. ; Hall, C.: Side Channel Cryptanalysis of Product Ciphers. In: *Computer Security - ESORICS 98*, Springer, 1998. – ISBN 978–3540650041, S. 97–110

[Len07] Lenze, Burkhard: Basiswissen Angewandte Mathematik: Numerik, Grafik, Kryptik, W3L GmbH, 2007. – ISBN 978–3937137827

[Man04] Mangard, S.: Hardware Countermeasures against DPA - A Statistical Analysis of Their Effectiveness. In: *Topics in Cryptology CT-RSA 2004*, Springer, 2004. – ISBN 978–3540209966, S. 222–235

[MBKP11] Moradi, A. ; Barenghi, A. ; Kasper, T. ; Paar, C.: On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks. In: *Proceedings of the 18th ACM conference on Computer and communications security* ACM, 2011. – ISBN 978–1450309486, S. 111–124

[Mes01] Messerges, T. S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: *Cryptographic Hardware and Embedded Systems—CHES 2000* Springer, 2001. – ISBN 978–3540414551, S. 238–251

[MHAS08] Miyamoto, A. ; Homma, N. ; Aoki, T. ; Satoh, A.: Enhanced Power Analysis Attack Using Chosen Message Against RSA Hardware Implementations. In: *Proceedings of 2008 IEEE International Symposium on Circuits and Systems* Institute of Electrical and Electronics Engineers, 2008. – ISBN 978–1424416837, S. 3282–3285

[MMSS02] Martino, B. ; Mazzocca, N. ; Saggese, G. P. ; Strollo, A. G.: A Technique for FPGA Synthesis Driven by Automatic Source Code Analysis and Transformations. In: *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream* Springer, 2002. – ISBN 978–3540441083, S. 47–58

[MO96] Menezes, A. J. ; v. Oorschot, P. C.: Handbook of Applied Cryptography, CRC Press, 1996. – ISBN 978–0849385230

[ÖOP03] Örs, S. B. ; Oswald, E. ; Preneel, B.: Power-Analysis Attacks on an FPGA - First Experimental Results. In: *Cryptographic Hardware and Embedded Systems-CHES 2003*, Springer, 2003. – ISBN 978–3540408338, S. 35–50

[Osw11] Oswald, E.: Einsatz und Bedeutung von Elliptischer Kurven für die elektronische Signatur Zentrum für sichere Informationstechnologie - Austria, 2011

[Paw98] Pawelczak, D.: Diplomarbeit: Entwurf und Realisierung eines Trust Centers, Universität München Fachbereich Informatik, 1998

[Plo14] Ploss, R.: Industrie 4.0 – Chance für Europas Wirtschaft. In: *e & i Elektrotechnik und Informationstechnik* Springer, 2014. – ISSN 1613–7620, S. 192

[Pog07] Poguntke, Werner: Basiswissen IT-Sicherheit: Das Wichtigste für den Schutz von Systemen und Daten, W3l, 2007. – ISBN 978–3937137650

[Rüd09] Rüdinger, J.: Auswirkungen von Seitenkanalangriffen auf das Design kryptographischer Algorithmen, Jörg Vogt Verlag, 2009. – ISBN 978–3938860274

[SLP05] Schindler, W. ; Lemke, K. ; Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: *Cryptographic Hardware and Embedded Systems-CHES 2005*, Springer, 2005. – ISBN 978–3540284741, S. 30–46

[SOSQ03] Standaert, F.-X. ; v. O. t. Oldenzeel, L. ; Samyde, D. ; Quisquater, J.-J.: Power Analysis of FPGAs: How Practical Is the Attack? In: *Field Programmable Logic and Application*, Springer, 2003. – ISBN 978–3540408222, S. 701–710

[SSH+08] Satoh, A. ; Shamir, A. ; Homma, N. ; Miyamoto, A. ; Aoki, T.: Collision-Based Power Analysis of Modular Exponentiation Using Chosen-Message Pairs. In: *Cryptographic Hardware and Embedded Systems-CHES 2008*, Springer, 2008. – ISBN 978–3540850526, S. 15–29

[SSK10] Shelake, V. G. ; Shinde, S. A. ; Kamat, R. K.: Unleash the System On Chip using FPGAs and Handel C Springer, 2010. – ISBN 978–9048181117

[Sta10] Standaert, F.: Introduction to Side-Channel Attacks. In: *Secure Integrated Circuits and Systems*, Springer, 2010. – ISBN 978–0387718279, S. 27–42

[Swe08] Swenson, C.: Modern Cryptanalysis: Techniques for Advanced Code Breaking, John Wiley & Sons, 2008. – ISBN 978–0470135938

[VY08] Velegalati, R. ; Yalla, P. S V V K.: Differential Power Analysis Attack on FPGA Implementation of AES, George Mason University publisher, 2008

[Wan00] Wang, Hsiu-Chiung: Speed Improvements for the RSA Encryption Method, Edinburgh Napier University, 2000

[WWM11] Witteman, M. F. ; van Woudenberg, J. G. J. ; Menarini, F.: Defeating RSA Multiply-Always and Message Blinding Countermeasures. In: *Topics in Cryptology - CT-RSA 2011*, Springer, 2011. – ISBN 978–3642190735, S. 77–88

[Zef07] Zefferer, T.: Forschungsbericht Side-Channel Analysen Zentrum für sichere Informationstechnologie - Austria, 2007

Internet Referenzen

[1] Atlys reference manual. http://www.digilentinc.com/Data/Products/ATLYS/Atlys_rm_V2.pdf Zuletzt online am: 12.11.2014.

[2] Atlys schematics. http://www.digilentinc.com/Data/Products/ATLYS/Atlys_C2_sch.pdf Zuletzt online am: 12.11.2014.

[3] Bürgerkarte. http://www.buergerkarte.at/experten-informationen.html Zuletzt online am: 24.10.2014.

[4] Developing tamper resistant designs with xilinx virtex-6 and 7 series fpgas. http://www.xilinx.com/support/documentation/application_notes/xapp1084_tamp_resist_dsgns.pdf Zuletzt online am: 27.10.2014.

[5] Evaluation environment for side-channel attacks. http://www.risec.aist.go.jp/project/sasebo/ Zuletzt online am: 21.11.2014.

[6] Spartan-6 family overview. http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf Zuletzt online am: 13.11.2014.

[7] Pressemitteilungen, Jänner 2010. http://www3.uni-bonn.de/Pressemitteilungen/004-2010 Zuletzt online am: 23.10.2014.