Fachhochschule Münster

Fachbereich Elektrotechnik und Informatik

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

im Studiengang Informatik

Angriff auf eine Implementierung des Verschlüsselungsverfahrens AES in Microcontrollern mittels Differential Power Analysis

- Entwurf, Aufbau und Erprobung einer Laborumgebung -

Erstprüfer Prof. Dr.-Ing. Sebastian Schinzel
Zweitprüfer Hendrik Schwartke

vorgelegt am 31. Juli 2017
von Erich Klundt

Abstract

Angriffe auf kryptographische eingebettete Systeme sind nichts Neues. Der Angriff über Seitenkanäle, wie zum Beispiel dem Stromverbrauch sind besonders erfolgversprechend, da oftmals während dem Entwurf eines Systems keine Schutzmechanismen dagegen ergriffen werden. Diese Angriffe werden als Power Analysis Attacks bezeichnet. Die Relevanz eines solchen Angriffs hängt stark mit dem Aufwand und der benötigten Ausstattung zusammen. In dieser Arbeit wird auf den Aufbau einer Laborumgebung eingegangen und anhand zweier Verfahren die Durchführung von Power Analysis Attacks aufgezeigt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
ALU	Arithmetic Logic Unit
ASCII	American Standard Code for Information Interchange
CMOS	Complementary Metal-Oxide-Semiconductor
CPA	Correlation Power Analysis
CUDA	Compute Unified Device Architecture
DES	Data Encryption Standard
DPA	Differential Power Analysis
DSS	Digital Signature Standard
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standard
GPU	Graphics Processing Unit
IC	Integrated Circuit
LSB	Least Significant Bit
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSB	Most Significant Bit
NIST	National Institute of Standards and Technology
NMOS	n-channel Metal-Oxide-Semiconductor
PMOS	p-channel Metal-Oxide-Semiconductor
RFC	Requests for Comments
RISC	Reduced Instruction Set Computer

SPA Simple Power Analysis
SRAM Static random-access memory
SSH Secure Shell
SSL Secure Sockets Layer

TLS Transport Layer Security

USB Universal Serial Bus

WLAN Wireless Local Area Network

Glossar

Peak Mit einem Peak ist ein erhöhter Messwert gemeint. Der Begriff wird in Verbindung mit Grafiken verwendet, in denen Messwerte als Graphen dargestellt werden. Ein Peak lässt sich in der Regel durch eine Spitze von den anliegenden Messwerten der Umgebung abheben.

Trace Ein Trace stellt eine Menge von Messwerten des Stromverbrauchs einer kryptographischen Operation dar. Die Menge der Messwerte ist abhängig von der Speichertiefe des Oszilloskops. Wird das arithmetische Mittel mehrerer Messungen gebildet, so wird dies ebenfalls als Trace bezeichnet.

1 Einleitung

Bereits seit geraumer Zeit nehmen Microcontroller in unserem Alltag einen immer höher werdenden Stellenwert ein und können als unabdingbar angesehen werden. Der erste Microcontroller reicht bis in das Jahr 1971 zurück, in dem Gary Boone und Michael Cochran den TMS 1000 entwickelt haben. Zu diesem Zeitpunkt wurde der TMS 1000 hauptsächlich in Taschenrechnern des Unternehmens Texas Instruments eingesetzt, später auch in einem Türgong-Modell und in Nachbildungen der sogenannten Berlin-Uhr, welche im Jahre 1975 entwickelt wurde.

Gegenwärtig finden Microcontroller in zahlreichen weiteren Bereichen ihre Anwendung, werden im Alltag jedoch kaum mehr wahrgenommen, da sie sich zumeist in eingebetteten Systemen befinden und vor dem Anwender verborgen bleiben. Typische Einsatzbereiche sind zum Beispiel Haushaltsgeräte wie Mikrowellen, Waschmaschinen oder Kaffeemaschinen, Unterhaltungselektronik wie Telefone oder Fernsehen und der Einsatz in einer Vielzahl von Steuergeräten. Da Microcontroller überdies auch oft in sicherheitskritischen Bereichen eingesetzt werden, beispielsweise in Alarmanlagen, Fernbedienungen oder Smartcards, müssen Sicherheitsaspekte beachtet werden, um einen ausreichenden Schutz vor möglichen Angreifern zu gewährleisten.

Deshalb sollten bei der Betrachtung möglicher Angriffsszenarien, vor allem Seitenkanalangriffe, nicht außer Acht gelassen werden. Diese Art von Angriffen beziehen sich stets auf Informationen, welche durch die physische Implementierung eines kryptographischen Systems preisgegeben werden. Der erste bekannte und veröffentlichte Angriff dieser Art, bezog sich auf die Rechenzeit eines Systems und wurde bereits im Jahr 1996 von Kocher vorgestellt [Koc96]. Darauf aufbauend gibt es inzwischen eine ganze Reihe weiterer Angriffe, die sich solche Seitenkanalinformationen zunutze machen.

1.1 Motivation

Der AES-Algorithmus ist ein weit verbreitetes Verschlüsselungsverfahren. Er wird unter anderem in Verbindung mit dem Transport Layer Security (TLS)-Protokoll, einem hybriden Verschlüsselungsprotokoll, verwendet und gilt als sicher. Aufgrund der weiten Verbreitung von AES, ist ein Angriff auf diesen äußerst praxisrelevant.

Bislang konnte der AES-Algorithmus allen bekannten Kryptoanalysen widerstehen. Typische Angriffe auf kryptographische Systeme und deren Verschlüsselungsalgorithmen beruhen deshalb in der Regel auf Fehlern in einer konkreten Implementierung, statt auf Fehlern in dem Algorithmus. Oftmals werden dabei physische Angriffe gegen ein kryptographisches System, bekannt als Seitenkanalangriffe, völlig außer Acht gelassen. Infolgedessen sind kryptographische Systeme häufig gegenüber Seitenkanalangriffen anfällig.

Deshalb ist ein Angriff auf AES per Seitenkanal für einen Angreifer interessant. Insbesondere Power Analysis Attacks, welche sich auf die Unterschiede im Stromverbrauch eines kryptographischen Systems beziehen, sind hierbei von großer Bedeutung. Solche Angriffe sind in der Regel passiv und können demnach ohne das Hinterlassen von Spuren durchgeführt werden. Die Differential Power Analysis (DPA) stellt dabei ein mächtiges Verfahren dar. Hiermit kann kryptographisches Schlüsselmaterial eines Systems extrahiert werden.

Ein Vorteil aus Sicht eines Angreifers ist die Problematik, dass sich Gegenmaßnahmen gegen diese Art von Angriffen stets nur auf eine konkrete Implementierung beziehen. Da AES weit verbreitet ist, stellt die DPA eine reale Gefahr dar. Es stellt sich die Frage nach dem Aufwand einer solchen DPA.

In dieser Arbeit wird daher der Aufbau einer Laborumgebung für Power Analysis Attacks beschrieben. Zudem wird eine DPA am Beispiel eines 8-Bit Microcontrollers des Herstellers Atmel durchgeführt. Hierbei wird der geheime AES Schlüssel einer sich auf dem Microcontroller befindlichen AES-Verschlüsselung extrahiert.

1.2 Angriffsszenario

Seitenkanalangriffe im Allgemeinen sind Angriffe auf kryptographische Systeme jeglicher Art. Hierbei kann es sich beispielsweise um eine Smartcard handeln. Es sind somit eine Vielzahl an verschiedenen Szenarien denkbar. Im Rahmen dieser Arbeit wird das Angriffsszenario jedoch auf ein konkretes Szenario eingeschränkt. Bei diesem Angriffsszenario soll ein Power Analysis Attack, konkret eine DPA, durchgeführt werden. Hierbei wird ein kryptographisches System angegriffen. Das kryptographische System wird dabei durch einen 8-Bit Microcontroller realisiert, welcher eine Verschlüsselung durchführt. Der kryptographische Algorithmus wird dabei als bekannt angenommen. Der Angreifer hat das Ziel, das geheime Schlüsselmaterial mittels einer DPA zu extrahieren. Er besitzt dabei physischen Zugriff auf das System und kann dieses mit verschiedenen Eingabedaten bedienen. Bei den Eingabedaten handelt es sich hierbei um verschiedene vom Angreifer frei wählbare Klartexte.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit unterteilt sich in fünf Teile:

(i) Im ersten Teil (Abschnitt 2) wird auf die grundlegenden Eigenschaften von Seitenkanalangriffen eingegangen. Es wird eine Klassifizierung vorgenommen und einige wenige Arten von Seitenkanalangriffen werden näher erläutert.

(ii) Der zweite Teil (Abschnitt 3) führt in den AES-Algorithmus ein. Hierbei wird lediglich ein grober Überblick über die Arbeitsweise des Algorithmus gegeben.

(iii) Im dritten Teil (Abschnitt 4) wird konkret auf die Power Analysis Attacks eingegangen. Hierbei werden sowohl benötigte Grundlagen, als auch Vorgehensweisen für die Durchführung solcher Angriffe beschrieben.

(iv) Der vierte Teil beschreibt direkte Angriffe auf Microcontroller mittels Power Analysis Attacks. In diesem Teil wird die verwendete Laborumgebung (Abschnitt 5) gezeigt und auf den Testaufbau (Abschnitt 6) eingegangen.

(v) Letztendlich wird im letzten Teil (Abschnitt 7) ein Fazit gezogen.

2 Seitenkanalangriffe

In der heutigen Zeit gibt es eine ganze Reihe an gängigen Algorithmen für die Verschlüsselung von sensiblen Daten. Zahlreiche dieser Verfahren haben sich inzwischen schon lange bewährt. Hierzu gehören zum Beispiel Verschlüsselungsverfahren, wie RSA oder AES. Es sind bislang keine Angriffe bekannt, weder auf den vollen AES-Algorithmus noch auf den vollen RSA-Algorithmus, mithilfe derer es möglich ist, mit realistischem Aufwand und in angemessener Zeit an das verwendete Schlüsselmaterial zu gelangen. Jedoch geben kryptographische Systeme oft selbst interessante Informationen preis, wodurch Rückschlüsse auf das Schlüsselmaterial möglich sind. Die Preisgabe der Informationen geschieht über sogenannte Seitenkanäle, welche den Grundbaustein für die Seitenkanalangriffe bilden.

2.1 Definition

Bei einem Seitenkanalangriff handelt es sich um einen Angriff auf ein kryptographisches System. Diese Art von Angriffen beziehen sich stets auf physikalische Messgrößen, welche durch das kryptographische System, in Abhängigkeit einer bestimmten Implementierung, verursacht werden. Diese Messgrößen bilden den sogenannten Seitenkanal. Typische Messgrößen sind zum Beispiel die Zeit, die elektromagnetische Abstrahlung oder der Stromverbrauch. Ein Seitenkanalangriff richtet sich somit nicht gegen das kryptographische Verfahren selbst, sondern vielmehr gegen eine konkrete Implementierung eines Algorithmus.

2.2 Klassifikation

Aufgrund von verschiedenen Merkmalen, lassen sich Seitenkanalangriffe in verschiedene Klassen unterteilen. Eine solche Klassifizierung kann anhand der in Tabelle 2.1 aufgeführten Kriterien erfolgen [ZF05, S. 8].

Ein wichtiges Kriterium ist der Grad der Kontrolle über das System. Dieser lässt sich in die beiden Klassen *aktiv* und *passiv* unterteilen. Bei einem aktiven Angriff beeinflusst ein Angreifer das Verhalten des Zielsystems direkt. Die daraus resultierenden Verhaltensänderungen des Systems werden gemessen. Hingegen wird

Kriterium	Klassen
Grad der Kontrolle über das System	Aktiv, Passiv
Art des Zugriffs	Invasiv, Semi-Invasiv, Nichtinvasiv
Verwendetes Analyseverfahren	einfache Seitenkanalanalyse, differentielle Seitenkanalanalyse

Tabelle 2.1: Kriterien, anhand derer eine Klassifizierung von Seitenkanalangriffen möglich ist.

bei einem passiven Angriff das Verhalten des Zielsystems nicht beeinflusst. Es findet kein Eingriff auf Operationen oder Abläufe des Systems statt. Das Zielsystem verhält sich hierbei genau so, als wenn kein Angriff stattfindet. [ZF05, S. 8]

Des Weiteren ist auch eine Klassifizierung bezüglich der Art des Zugriffs durchführbar. Mögliche Klassen sind hierbei *Invasiv*, *Semi-Invasiv* und *Nichtinvasiv*. Invasive Angriffe benötigen direkten Zugriff auf die internen Komponenten des anzugreifenden kryptographischen System. Ein möglicher Invasiver Angriff wäre zum Beispiel das Zerstören der physischen Schicht eines Microcontrollers, um somit durch Microprobing[1] Zugriff auf den Datenbus zu erlangen. Auch die Semi-Invasiven Angriffe benötigen Zugriff auf das System. Diese Art von Angriff hat jedoch keine Zerstörung des Kryptosystems zur Folge. Zum Beispiel besteht die Möglichkeit, mithilfe von Laserstrahlen Einfluss auf den internen Zustand des Systems zu nehmen. Bei den Nichtinvasiven Angriffen handelt es sich um eine reine Beobachtung des Zielsystems. Diese Art von Angriff verwendet lediglich extern verfügbare Informationen. Daraus ergibt sich die besondere Eigenschaft, dass das Zielsystem selbst solche Angriffe nicht erkennen kann. So kann ein Microcontroller zum Beispiel nicht erkennen, ob gerade eine Messung des Stromverbrauchs durchgeführt wird oder nicht. [And+05, S. 6 f.]

Auch kann man Seitenkanalangriffe anhand des verwendeten Analyseverfahrens in verschiedene Klassen unterteilen. Man unterscheidet zwischen *einfachen Seitenkanalanalysen* und *differentiellen Seitenkanalanalysen*. Die einfache Seitenkanalanalyse ist ein Verfahren, welches Schlüsselabhängigkeiten ausnutzt. Hierbei sind die ausgeführten Operationen des kryptographischen Systems direkt abhängig von dem verwendeten Schlüssel. In der Regel reicht bei diesem Verfahren eine einzige Messung aus, um an geheimes Schlüsselmaterial zu gelangen. Hingegen beruht die differentielle Seitenkanalanalyse auf Messungen mit bekannten, jedoch variablen Daten. Es müssen deshalb mehrere Messungen erfolgen. Die Auswertung von diesen durchgeführten Messungen erfolgt dann mittels statistischer Verfahren. [ZF05, S. 9]

[1] Unter Microprobing versteht man das Untersuchen von einzelnen Leitungen oder Speicherzellen mittels spezieller Hardware.

2.3 Angriffstypen

Durch die unterschiedlichen physikalischen Messgrößen, welche das kryptographische System verursacht, ergeben sich eine Vielzahl an verschiedenen Angriffsmöglichkeiten. In den nachfolgenden Abschnitten befindet sich eine kurze Beschreibung von drei weit verbreiteten Angriffsarten.

2.3.1 Timing Attack

Timing Attacks nutzen die unterschiedliche Rechenzeit, die ein kryptographisches System während der Ausführung eines Algorithmus benötigt, aus. Die Laufzeit einer kryptographischen Operation ist nicht nur von dem System selbst oder dem gewählten Algorithmus abhängig, sondern hängt oftmals sowohl von den Eingaben als auch dem verwendeten Schlüsselmaterial ab. Hierdurch kann man messbare Unterschiede in der Rechenzeit verzeichnen.

Als Erfinder der Timing Attacks gilt Kocher. Er befasste sich im Jahr 1996 [Koc96] mit dem Thema der Seitenkanalangriffe. Schon zu dieser Zeit zeigte er auf, dass ein Angreifer durch eine solche Laufzeitanalyse an wertvolle kryptographische Informationen gelangen kann. Dies umfasst zum Beispiel das verwendete Schlüsselmaterial. Er bezog sich hierbei unter anderem auf Implementierungen von Diffie-Hellman, RSA und dem Digital Signature Standard (DSS).

2.3.2 Electromagnetic Attack

Auch die *Electromagnetic Attacks* bieten eine Möglichkeit, um an relevante Schlüsselinformationen eines kryptographischen Systems zu gelangen. Die meisten elektronischen Geräte geben elektromagnetische Strahlung ab. Das hierdurch erzeugte elektromagnetische Feld lässt sich messen und verrät ebenfalls einiges über das kryptographische System. Hiermit kann ein Angreifer ebenso an geheime Informationen gelangen.

2.3.3 Power Analysis Attack

Die *Power Analysis Attacks* befassen sich mit der Messung des Stromverbrauches eines kryptographischen Systems, während dem Ausführen eines Algorithmus. Mit Hilfe dieser Art von Angriffen können durch das Messen des Stromverbrauchs Rückschlüsse auf das verwendete Schlüsselmaterial gezogen werden. Dies resultiert daher, dass der Stromverbrauch eines kryptographischen Systems nicht konstant ist, sondern unter anderem von der gerade ausgeführten Operation beeinflusst wird. Kocher, Jaffe und Jun haben im Jahr 1999 [KJJ99] erste Verfahren vorgestellt, um erfolgreiche Power Analysis Attacks durchführen zu können. Es wurden die beiden Verfahren Simple

Power Analysis (SPA) und DPA beschrieben und gezeigt, wie damit der verwendete Schlüssel des Verschlüsselungsverfahren Data Encryption Standard (DES) ermittelt werden kann. In Abschnitt 4.6 wird noch näher auf diese Verfahren eingegangen.

2.4 Stand der Forschung

Es gibt bereits zahlreiche bekannte Seitenkanalangriffe, die sich auf die Implementierung verschiedener Algorithmen beziehen. In diesem Abschnitt werden einige Angriffe gegen die in Abschnitt 2.3 aufgeführten Arten von Seitenkanalangriffen aufgezeigt und näher erläutert.

Bereits im Jahr 2003 haben Brumley und Boneh gezeigt, wie der private Schlüssel einer Secure Sockets Layer (SSL)-Verschlüsselung extrahiert werden kann [BB03]. Zum Einsatz kam hierbei ein Server mit OpenSSL, eine SSL-Bibliothek, die oftmals auf Webservern eingesetzt wird. Durch einen Timing Attack haben Brumley und Boneh den privaten Schlüssel innerhalb von ca. 2 Stunden ermittelt. Hierfür wurden von einem Client Anfragen an den Server geschickt und die Zeit, die der Server für das Entschlüsseln benötigte, gemessen.

Ferner haben Genkin u. a. einen Angriff auf die Elliptische-Kurven-Kryptografie vorgestellt [Gen+16a]. Konkret wurde das Elliptic Curve Diffie-Hellman (ECDH) Verfahren mittels eines Electromagnetic Attack angegriffen und so der geheime Schlüssel ermittelt. Der Angriff bezog sich auf die Implementierung des ECDH Algorithmus der Libgcrypt Bibliothek, die auf einem Notebook ausgeführt wurde. Durch das Messen der elektromagnetischen Abstrahlung konnte der geheime Schlüssel innerhalb von wenigen Sekunden herausgefunden werden. Zudem wurde 2016 ein weiterer Angriff auf die Elliptische-Kurven-Kryptografie vorgestellt [Gen+16b]. Es wurde gezeigt, dass Smartphones ebenfalls anfällig gegenüber Electromagnetic Attacks sind. Der dort durchgeführte Angriff richtete sich gegen die Elliptic Curve Digital Signature Algorithm (ECDSA) Implementierung von OpenSSL.

Ebenso gibt es bekannte Power Analysis Attacks auf Systeme. Hierbei sollte das Keeloq System erwähnt werden. Bei Keeloq handelt es sich um ein Zugangsberechtigungssystem, das insbesondere bei Autos zur Türöffnung oder bei Garagentoren eingesetzt wird. 2008 wurde bekannt gegeben, dass das Verschlüsselungssystem durch einen Power Analysis Attack gebrochen wurde [Eis+08]. Den Wissenschaftlern war es gelungen, sowohl den geheimen Schlüssel eines Senders, als auch den Herstellerschlüssel des Empfängers aufzudecken.

3 AES

Bei dem Advanced Encryption Standard (AES) handelt es sich um ein symmetrisches Verschlüsselungsverfahren. AES ist ebenfalls unter dem Namen *Rijndael* bekannt, welcher sich aus den Namen der beiden Entwickler Joan Daemen und Vincent Rijmen ableiten lässt. Im Jahre 2001 erklärte das National Institute of Standards and Technology (NIST) AES als Federal Information Processing Standard (FIPS). Da AES ein symmetrisches Verschlüsselungsverfahren ist, findet sowohl die Verschlüsselung als auch die Entschlüsselung mit dem gleichen Schlüssel statt. Hierdurch muss jedes System, das mit AES verschlüsselten Daten arbeitet, Kenntnis von dem festgelegten Schlüssel haben. Es handelt sich um eine Blockchiffre, sodass stets einzelne Blöcke mit einer festgelegten Länge verarbeitet werden. Die Länge eines zu verschlüsselnden Blockes beträgt 128 Bit, wohingegen variable Schlüssellängen von 128, 192 oder 256 Bit zulässig sind. Für die Verwendung der verschiedenen Schlüssellängen sind die Bezeichnungen AES-128, AES-192 und AES-256 üblich. Im Laufe der AES Verschlüsselung werden Transformationen auf jeden einzelnen Block durchgeführt. Hierbei handelt es sich unter anderem um Transformationen wie das Substituieren oder das Vertauschen einzelner Bytes.

3.1 Ablauf

Da AES auf Blöcken mit einer festgelegten Blockgröße von 16 Bytes operiert, kann man die einzelnen Blöcke der AES-Verschlüsselung als 4×4-Matrix der Form

$$S = \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix}$$

auffassen. Diese Matrix wird in der Regel als *State* S bezeichnet. Verschiedene Operationen, welche in Abschnitt 3.4 näher beschrieben werden, können anschließend mehrmalig auf den State angewendet werden, um diesen zu transformieren. Jede dieser Iterationen wird als eine Runde aufgefasst, wobei die Anzahl der Runden von der Schlüssellänge abhängig ist. Tabelle 3.1 gibt Aufschluss über die Anzahl der

Schlüssellänge k (bit)	Anzahl der Runden R
128	10
192	12
256	14

Tabelle 3.1: Anzahl der AES Runden in Abhängigkeit zur gewählten Schlüssellänge. Die Anzahl der Runden steigt mit der Länge des Schlüssels.

Runden R in Abhängigkeit zur gewählten Schlüssellänge k.

In Abbildung 3.1 ist der Ablauf einer Verschlüsselung mit AES zu sehen. Zu erkennen sind die Vorrunde, die $R-1$ Verschlüsselungsrunden und die Schlussrunde, welche in der Rundenanzahl R mitgezählt wird. Die Entschlüsselung verläuft analog zur Verschlüsselung. Hierbei müssen jedoch alle Operationen invertiert und in umgekehrter Reihenfolge durchgeführt werden.

3.2 S-Box

Eine zentrale Komponente bei der AES-Verschlüsselung ist die S-Box (engl. *substitution box*). Diese dient der Konfusion. Grundlegend kann man S-Boxen dynamisch, in Abhängigkeit des gewählten Schlüssels erzeugen oder statisch festlegen. AES verwendet eine statisch festgelegte S-Box. Sie kann als Lookup-Tabelle definiert werden und bietet die Möglichkeit, ein Byte durch ein anderes Byte zu ersetzen. Da AES auf einzelnen Bytes operiert, enthält die S-Box 256 verschiedene Werte. Es gilt zu beachten, dass bei einer Entschlüsselung auch die S-Box invertiert werden muss. Die S-Box ist somit bijektiv. In Anhang A.2 kann die vollständige S-Box des AES eingesehen werden.

3.3 KeyExpansion

Bei der AES-Verschlüsselung wird für jede einzelne Runde ein sogenannter Rundenschlüssel benötigt. Das Erstellen dieser Teilschlüssel wird häufig als *KeyExpansion* bezeichnet und ist ebenfalls ein wesentlicher Bestandteil der AES-Verschlüsselung. Die einzelnen Rundenschlüssel sind hierbei abhängig von dem initial gewählten Schlüssel. Dabei entspricht die Länge der Rundenschlüssel immer der Blockgröße. Demnach sind die erzeugten Rundenschlüssel 16 Bytes lang.

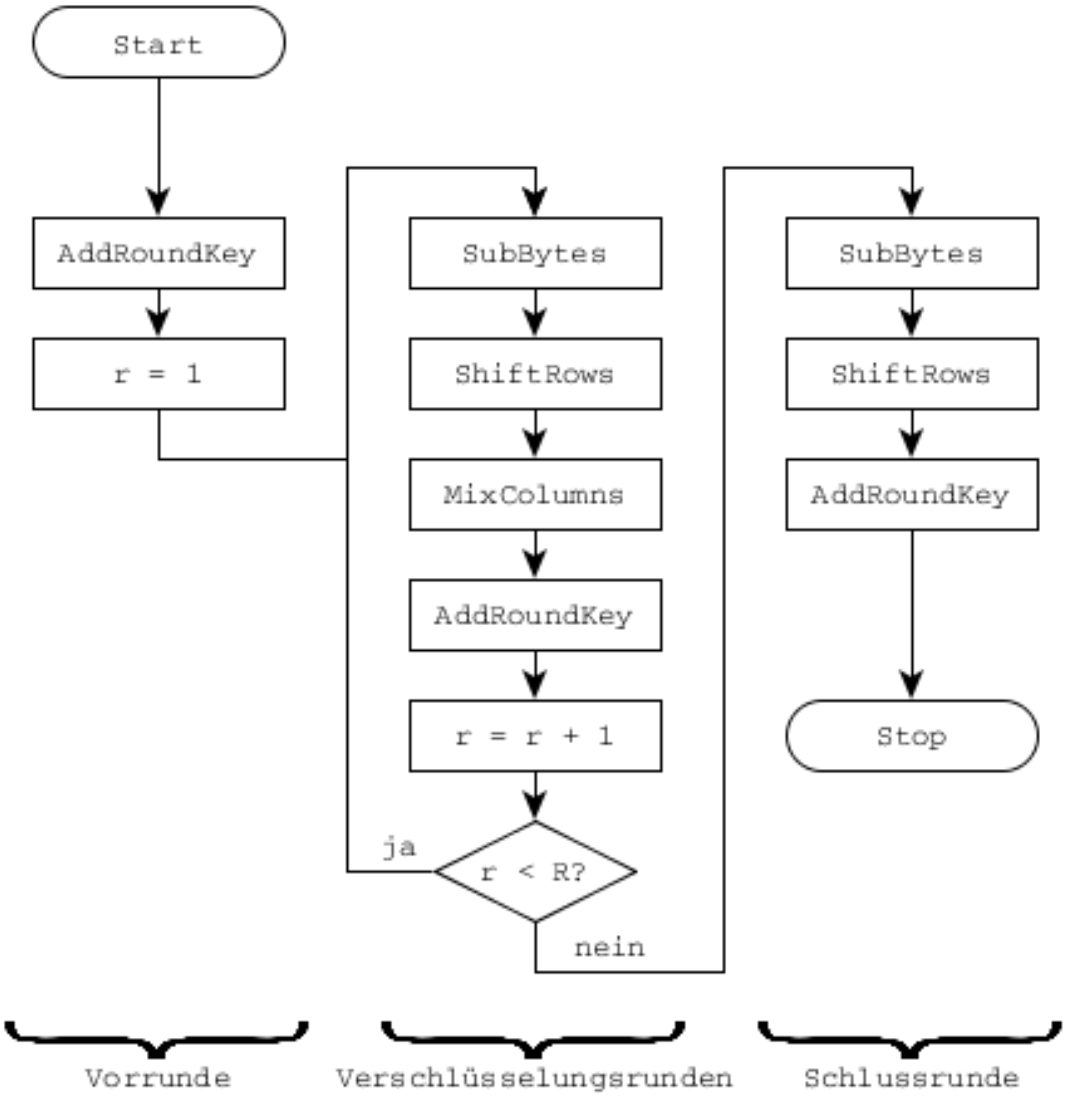


Abbildung 3.1: Schematische Darstellung des Ablaufs einer AES Verschlüsselung. R entspricht der Anzahl der Runden (10, 12 oder 14). Diese ist abhängig von der Länge des gewählten Schlüssels (siehe Tabelle 3.1).

3.4 Operationen

Die folgenden Abschnitte beschreiben die verschiedenen Operationen des AES. Hierbei wird lediglich auf die Arbeitsweise der einzelnen Operationen eingegangen, jedoch nicht auf die zugrundeliegende mathematische Theorie.

3.4.1 SubBytes

Die *SubBytes* Operation führt eine monoalphabetische Substitution jedes einzelnen Bytes b_{ij} der State Matrix, unter Zuhilfenahme der S-Box, durch:

$$\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \xmapsto{b'_{ij} = SBox(b_{ij})} \begin{pmatrix} b'_{0,0} & b'_{0,1} & b'_{0,2} & b'_{0,3} \\ b'_{1,0} & b'_{1,1} & b'_{1,2} & b'_{1,3} \\ b'_{2,0} & b'_{2,1} & b'_{2,2} & b'_{2,3} \\ b'_{3,0} & b'_{3,1} & b'_{3,2} & b'_{3,3} \end{pmatrix}$$

Jedes Byte der State Matrix wird demnach durch ein anderes Byte ersetzt.

3.4.2 ShiftRows

In der *ShiftRows* Operation werden die Zeilen der State Matrix um eine bestimmte Anzahl von Spalten nach links rotiert:

$$\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \longmapsto \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{pmatrix}$$

Die Anzahl der Spalten, um die rotiert wird, ist abhängig von der Zeile. Hierbei wird Zeile i um genau i Spalten rotiert.

3.4.3 MixColumns

Mit Hilfe der Verknüpfung

$$\begin{aligned} b_{0,i} &= (a_{0,i} \cdot 2) \oplus (a_{1,i} \cdot 3) \oplus (a_{2,i} \cdot 1) \oplus (a_{3,i} \cdot 1) \\ b_{1,i} &= (a_{0,i} \cdot 1) \oplus (a_{1,i} \cdot 2) \oplus (a_{2,i} \cdot 3) \oplus (a_{3,i} \cdot 1) \\ b_{2,i} &= (a_{0,i} \cdot 1) \oplus (a_{1,i} \cdot 1) \oplus (a_{2,i} \cdot 2) \oplus (a_{3,i} \cdot 3) \\ b_{3,i} &= (a_{0,i} \cdot 3) \oplus (a_{1,i} \cdot 1) \oplus (a_{2,i} \cdot 1) \oplus (a_{3,i} \cdot 2) \end{aligned}$$

werden in der *MixColumns* Operation die Bytes $a_{0,i}, \ldots, a_{3,i}$ jeder einzelnen Spalte i der State Matrix miteinander vermischt. $b_{0,i}, \ldots, b_{3,i}$ entspricht hierbei der neuen Spalte.

3.4.4 AddRoundKey

Die *AddRoundKey* Operation führt eine bitweise XOR-Verknüpfung zwischen den Bytes der State Matrix und dem jeweiligen Rundenschlüssel durch. Bei dieser Operation handelt es sich um die einzige Funktion, die direkt von dem gewählten Schlüssel, beziehungsweise den daraus erzeugten Rundenschlüsseln, abhängig ist.

3.5 Anwendungsgebiete

Heutzutage ist die Anwendung kryptographischer Verfahren weit verbreitet. So ist AES in den USA als Verschlüsselungsverfahren für die Geheimhaltung staatlicher Informationen zugelassen [Nat03]. Hierbei dürfen AES-128, AES-192 und AES-256 bis zur Geheimhaltungsstufe *Secret* verwendet werden. Für die Geheimhaltungsstufe *Top Secret* sind hingegen lediglich Schlüssellängen von 192 oder 256 Bit (AES-192, AES-256) erlaubt.

Des Weiteren findet AES auch in dem Verschlüsselungsstandard IEEE 802.11i, welcher sich mit der Sicherheit von Wireless Local Area Network (WLAN) auseinandersetzt, seine Anwendung. Zudem kommt AES bei der Kommunikation per Secure Shell (SSH) zum Einsatz. Mit Hilfe von SSH kann eine gesicherte Netzwerkverbindung zwischen zwei Geräten aufgebaut werden.

Es gibt zahlreiche weitere Bereiche in denen AES verwendet wird. AES ist in x86-Prozessoren oftmals bereits direkt in Hardware implementiert, sodass spezielle Assemblerbefehle für das Verschlüsseln und Entschlüsseln existieren. Die Erweiterung ist unter dem Namen Advanced Encryption Standard New Instructions (AES-NI) bekannt. Hierdurch wird das Verschlüsseln und Entschlüsseln beschleunigt, sodass AES vermehrt auch auf ARM-Prozessoren, wie zum Beispiel in mobilen Geräten beziehungsweise eingebetteten Systemen, Anwendung findet.

3.6 Bekannte Angriffe

Laut aktuellem Stand gilt AES als sicheres Verschlüsselungsverfahren. Bislang sind keine praxisrelevanten Angriffe gegen AES bekannt, mit Hilfe derer der verwendete AES Schlüssel in angemessener Zeit ermittelt werden kann. Es existieren jedoch einige theoretische Angriffe, die einen Angriff auf den AES-Algorithmus ermöglichen.

Interessant ist hierbei jedoch lediglich der von Bogdanov, Khovratovich und Rechberger vorgestellte Biclique-Angriff [BKR11]. Der Biclique-Angriff ist im Vergleich zu einem Brute-Force-Angriff auf den AES-Algorithmus ca. um den Faktor 3 bis 5 mal schneller. Das heißt der Schlüssel kann bei AES-128 in $2^{126,1}$ Schritten, bei AES-192 in $2^{189,7}$ Schritten und bei AES-256 in $2^{254,4}$ Schritten berechnet werden. Da dies jedoch dennoch nicht in angemessener Zeit machbar ist, hat dieser Angriff in der Praxis bislang keinen hohen Stellenwert.

Biryukov und Großschädl haben sich im Jahr 2011 der Frage gewidmet, wie hoch die Kosten für den Bau eines Supercomputers für einen Angriff auf den vollen AES-Algorithmus sind. Das Ergebnis ist in einem Paper zusammengefasst [BG11]. Für den Supercomputer wurden 1 Billion US-Dollar veranschlagt, der Stromverbrauch beläuft sich auf 4 Terawatt. Theoretisch scheint der Bau eines solchen Supercomputers tatsächlich möglich zu sein. In der Realität wird es hierzu hingegen vermutlich nicht kommen. Es liegt nahe, dass der AES somit vorerst weiterhin als sicher gilt.

Es existieren einige weitere theoretische Angriffe auf den AES-Algorithmus. Diese beziehen sich jedoch nicht auf den vollen AES-Algorithmus, sondern nehmen Einschränkungen vor. So richten sich diese Angriffe zumeist gegen Varianten des AES-Algorithmus, bei denen die Anzahl der Runden reduziert wird. All diese Angriffe stellen demnach in der Praxis keine wirkliche Bedrohung für AES dar.

3.7 Seitenkanalangriff

Aus dem vorhergehenden Abschnitt geht hervor, dass AES auf herkömmlichen Wege bislang nicht gebrochen ist. Seitenkanalangriffe, vor allem Power Analysis Attacks, bieten jedoch einen anderen Angriffsweg.

Angreifbar sind hierdurch insbesondere AES Implementierungen auf eingebetteten Systemen. AES verwendet den geheimen Schlüssel byteorientiert, führt Operationen somit sequenziell auf einzelne 8 Bit durch. Durch eine DPA kann der Schlüssel einer solchen AES Implementierung ermittelt werden.

Eine typische Kryptoanalyse per Brute-Force-Methode würde den geheimen Schlüssel bei AES-128 in 2^{128} Schritten, bei AES-192 in 2^{192} Schritten und bei AES-256 in 2^{256} Schritten berechnen. Durch eine DPA kann der Suchraum deutlich reduziert werden. Da die DPA sich auf einzelne Bytes des geheimen Schlüssels bezieht, sind nur noch 2^8 Kombinationen pro Byte möglich. Bezogen auf AES-128, also einer Schlüssellänge von 16 Byte, muss nochmals mit 16 multipliziert werden. Für AES-192 muss also mit 24 und für AES-256 mit 32 multipliziert werden. Da die DPA auf mehrere Messpunkte angewendet wird, muss zusätzlich noch die Anzahl der Messpunkte mit berechnet werden. Der Suchraum der Kryptoanalyse reduziert sich somit auf

$2^8 \cdot 16 \cdot x$, wobei x der Anzahl der Messpunkte entspricht. Die DPA stellt im Vergleich zur Brute-Force-Methode ein geeignetes Angriffsverfahren dar.

4 Power Analysis Attacks

Power Analysis Attacks stellen eine wichtige Art von Seitenkanalangriffen dar. Mit Hilfe dieser kann geheimes Schlüsselmaterial eines kryptographischen Systems ermittelt werden. Dieser Abschnitt gibt einen Überblick über die typischen Komponenten und vermittelt Grundlagen. Des Weiteren werden zwei Verfahren vorgestellt, nämlich die SPA und die DPA, welche in den Abschnitten 6.5 und 6.6 verwendet werden. Zudem wird auf mögliche Gegenmaßnahmen eingegangen.

4.1 Überblick

In Abbildung 4.1 ist der grundlegende Aufbau bei der Durchführung eines Power Analysis Attacks zu sehen. Der Aufbau umfasst im wesentlichen drei Komponenten.

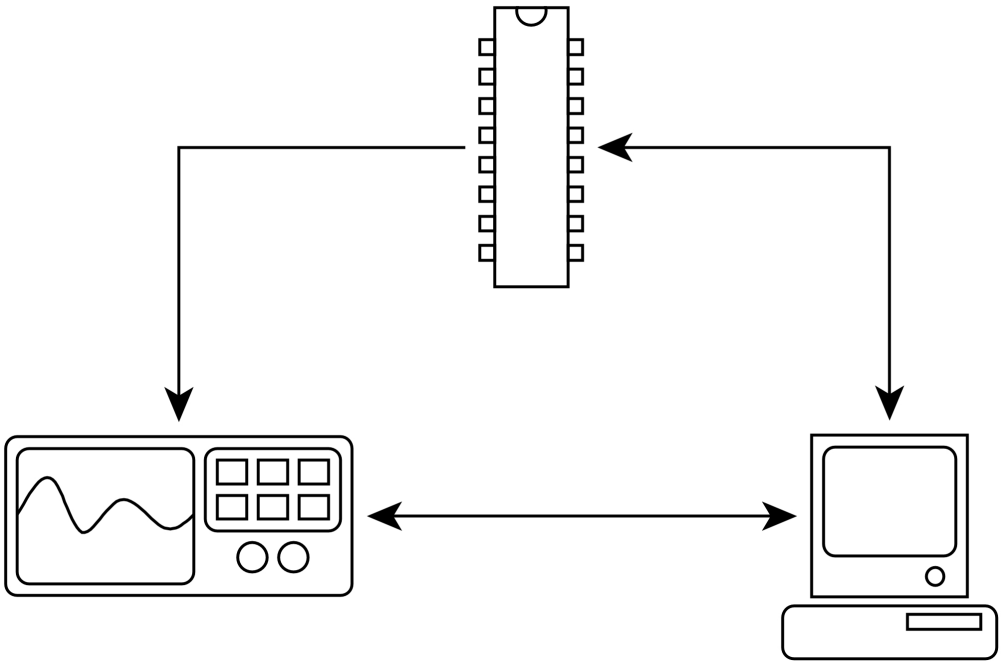


Abbildung 4.1: Die benötigten Komponenten für die Durchführung von Power Analysis Attacks umfassen im wesentlichen einen Computer, ein Oszilloskop und das anzugreifende kryptographische System. In dieser Arbeit handelt es sich bei dem anzugreifenden System um einen Microcontroller.

Hierbei handelt es sich um einen Computer, ein Oszilloskop und das anzugreifende kryptographische System. Für automatisierte Angriffe muss eine Kommunikation beziehungsweise ein Datenaustausch zwischen diesen drei Komponenten möglich sein.

Zum einen muss eine automatisierte Ansteuerung des kryptographischen Systems, durch den Computer, möglich sein. Dies bedeutet, dass sowohl die kryptographische Operation, die auf dem Kryptosystem läuft, automatisiert ausführbar sein sollte, als auch die Verwendung mit verschiedenen Daten möglich sein muss. Bei diesen Daten handelt es sich im Allgemeinen um den Klartext oder das Chiffrat eines Verschlüsselungsverfahrens.

Ferner ist die Kommunikation zwischen dem Computer und dem Oszilloskop wichtig. Die vom Oszilloskop aufgezeichneten Messungen sollten automatisiert an den Computer übertragbar sein. Ein Oszilloskop mit einer entsprechenden Schnittstelle, die so etwas anbietet ist somit von Vorteil.

Zum anderen müssen die Messungen des Oszilloskops und die kryptographische Operation des anzugreifenden Systems synchronisiert werden können. Genauer gesagt muss die Möglichkeit bestehen, den Stromverbrauch des anzugreifenden Systems während der Ausführung der kryptographischen Operation zu messen. Wie genau eine solche Messung durchzuführen ist, wird in Abschnitt 4.2.3 näher erläutert.

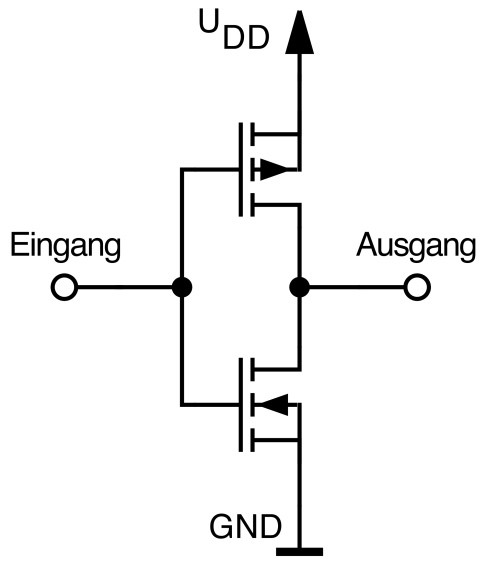
4.2 Grundlagen

Für die Durchführung von Power Analysis Attacks ist ferner ein grundlegendes Wissen in Bezug auf den Aufbau des Microcontrollers oder der dahinterstehenden Technologie des Datenbus erforderlich. Deshalb gehen Abschnitt 4.2.1 und Abschnitt 4.2.2 näher auf die genannten Themen ein. Zudem wird in Abschnitt 4.2.3 auf weitere benötigte Grundlagen in Bezug auf die Durchführung der Messung des Stromverbrauchs eingegangen.

4.2.1 CMOS-Technologie

In diesem Abschnitt wird ein Grundverständnis für die Complementary Metal-Oxide-Semiconductor (CMOS)-Technologie vermittelt. Bei der CMOS-Technologie handelt es sich um einen Schaltkreis bestehend aus Transistoren. Als Transistoren kommen hierbei Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs) zum Einsatz, und zwar paarweise sowohl n-channel Metal-Oxide-Semiconductor (NMOS)- als auch p-channel Metal-Oxide-Semiconductor (PMOS)-Transistoren. Heutzutage ist die CMOS-Technologie sehr weit verbreitet und wird vorwiegend in Integrated Circuits (ICs) verwendet.

Abbildung 4.2 zeigt einen einfachen Inverter in CMOS-Technik. Dieser kann aus zwei Transistoren, einem n-Kanal Transistor und einem p-Kanal Transistor, erstellt werden. Wie in der Abbildung zu sehen, besitzen MOSFETs drei Anschlüsse: *Gate (G)*, *Drain (D)* und *Source (S)*. Durch das Anlegen einer Spannung an dem Gate Anschluss wird der Stromfluss und somit der Widerstand auf der Strecke zwischen Drain und Source beeinflusst. Der MOSFET wirkt somit wie ein spannungsgesteuerter Widerstand. Im geöffnetem Zustand ist dieser hochohmig, im geschlossen hingegen niederohmig.



Bildquelle: [Coma]

Abbildung 4.2: Ein Inverter in CMOS-Technik, bestehend aus einem PMOS und einem NMOS.

4.2.2 Aufbau Microcontroller

Die zugrundeliegende Beschreibung über den Aufbau eines Microcontrollers gibt einen Überblick über die allgemeine Architektur. Als Referenz dienen der ATmega48A/PA, ATmega88A/PA, ATmega168A/PA und der ATmega328/P. Hierbei handelt es sich um 8-Bit Microcontroller der Atmel AVR Familie.

In Anhang A.3 ist die Architektur graphisch abgebildet. Eine wichtige Einheit stellen die Register dar. Bei diesen handelt es sich z.B. um Register mit kurzer Zugriffszeit, in denen Daten gespeichert werden können. Die Arithmetic Logic Unit (ALU), welche für arithmetische und logische Operation zuständig ist, gehört ebenfalls zum Kern der Architektur. Zudem sind die verschiedenen Speicherbausteine wie der Flash-Speicher, der Electrically Erasable Programmable Read-Only Memory (EEPROM) und der Static random-access memory (SRAM) weitere wichtige Einheiten. All diese Einheiten kommunizieren über einen Datenbus miteinander. Dieser hat eine Busbreite von 8 Bit. Es können somit parallel jeweils 8 Bit verarbeitet werden.

Für das Übertragen eines Wert aus einem Register zur ALU wird somit zwingend der Datenbus benötigt. Das heißt die einzelnen Leitungen des Datenbus müssen verändert, also auf logisch „1“ oder logisch „0“ gezogen werden. Auf physischer Ebene werden die Veränderungen des Datenbus hierfür mit Hilfe von elektronischen Schaltern bewerkstelligt. Hierbei kommt die in Abschnitt 4.2.1 beschriebene CMOS-Technologie zum Einsatz. Diese Veränderungen bewirken, dass beim Schalten, demnach bei einem Übergang von $0 \rightarrow 1$ oder von $1 \rightarrow 0$, ein Stromfluss stattfindet. Bei Übergängen

von $0 \rightarrow 0$ und von $1 \rightarrow 1$ hingegen fließt nur ein geringer Strom.

4.2.3 Durchführung der Messung

Die Durchführung der Messung kann auf zwei unterschiedliche Arten erfolgen. Die beiden Arten unterscheiden sich lediglich darin, an welcher Stelle im Stromkreislauf die Messung des Stromverbrauchs durchgeführt wird. So kann die Messung entweder an der Masseleitung oder direkt innerhalb der Spannungsversorgung durchgeführt werden. Die Messung sollte hierbei möglichst nah am anzugreifenden System, zum Beispiel an einem Microcontroller, realisiert werden. [KJJ99, S. 389]

In Abbildung 4.3 ist gezeigt, wie ein solcher Messaufbau aussehen kann. Für die

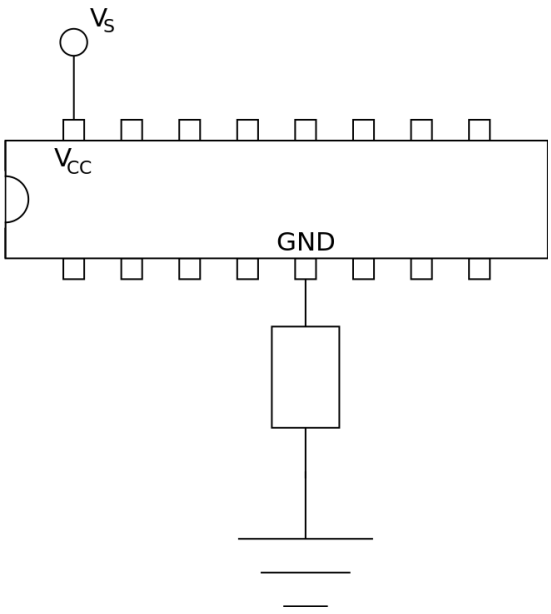


Abbildung 4.3: Messaufbau für die Durchführung der Messung des Stroms. In diesem Beispiel wird der Strom direkt an der Masseleitung gemessen.

direkte Messung an der Masseleitung wird ein Widerstand zwischen dem GND Pin des anzugreifenden Systems und GND der Spannungsversorgung (z.B. einem Netzgerät) eingesetzt. Alternativ kann der Widerstand, für die direkte Messung in der Spannungsversorgung, auch zwischen V_{CC} und V_S eingesetzt werden. Bei dem Widerstand sollte ein möglichst niederohmiger Widerstand, wie zum Beispiel 50 Ohm, gewählt werden.

Die eigentliche Messung erfolgt mittels des Oszilloskops. Hierfür wird mit Hilfe eines Tastkopfs der Spannungsabfall über dem eingebauten Widerstand gemessen.

Dabei sollte der Kanal des Tastkopfs AC gekoppelt sein, wodurch der Gleichspannungsanteil des Signals unterdrückt wird und lediglich die Spannungsänderungen sichtbar sind. Durch das ohmsche Gesetz kann daraufhin die Stromstärke berechnet werden. Da sowohl die Spannung als auch der eingebaute Widerstand konstant sind, verhält sich die Stromstärke proportional zum gemessenen Spannungsabfall über den Widerstand. Deshalb wird bei Power Analysis Attacks häufig die eigentliche Stromstärke erst gar nicht berechnet, sondern stattdessen direkt mit den gemessenen Spannungswerten gearbeitet. Dies ist auch der Grund, weswegen in Grafiken oftmals der Spannungsabfall dargestellt wird, in ihren Beschreibungen jedoch trotz alledem von einem Stromverbrauch gesprochen wird.[1]

4.3 Prinzip

Ein Microcontroller benötigt während seiner Ausführung Strom. Hierbei handelt es sich nicht nur um einen konstanten, sondern auch um einen variablen Stromverbrauch. Das Prinzip der Power Analysis Attacks stützt sich auf den variablen Stromverbrauch, genauer auf die folgenden zwei Umstände. Zum einen variiert der Stromverbrauch des Microcontrollers in Abhängigkeit von der ausgeführten Operation, zum anderen haben auch die zu verarbeitenden Daten Einfluss auf den Stromverbrauch.

Abbildung 4.4 stellt diesen Sachverhalt graphisch dar. Zu sehen ist der gemessene

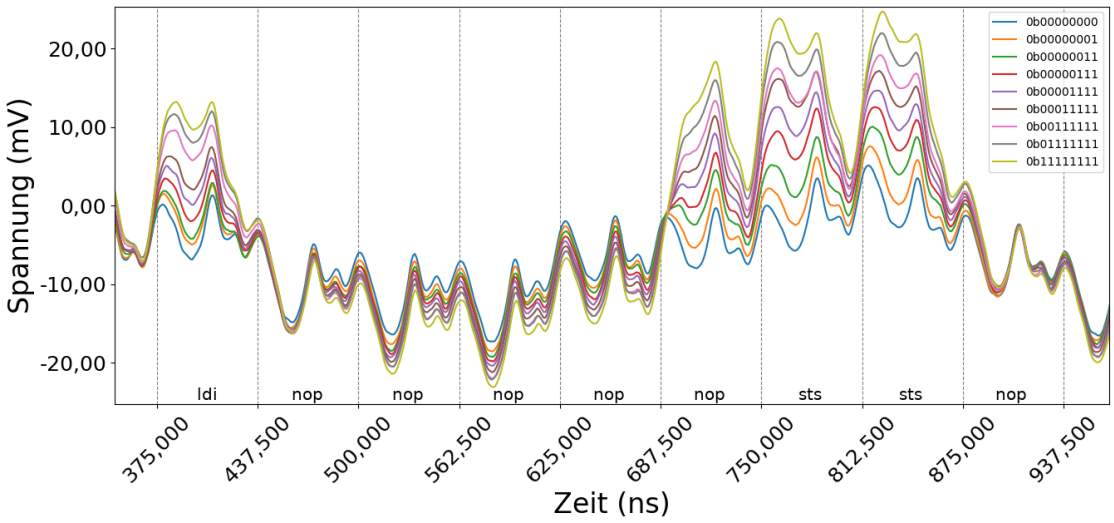


Abbildung 4.4: Messungen des Stromverbrauchs bei dem Laden und Speichern verschiedener Werte. Es sind deutliche Unterschiede zu erkennen. Je mehr Bitübergänge stattfinden, desto höher der Stromverbrauch. Die Messungen erfolgten auf einem 8-Bit Microcontroller.

Stromverbrauch eines auf einem Microcontroller ausgeführten Programms. Hierfür

[1] Dies entspricht ebenfalls den in dieser Arbeit abgebildeten Grafiken.

wurde die in Abschnitt 5 beschriebene Laborumgebung verwendet. Insgesamt wurden Messungen für neun unterschiedliche Daten durchgeführt. Um ein genaueres Ergebnis zu erzielen, stellt der Graph das arithmetische Mittel von 100 durchgeführten Messungen dar. Das ausgeführte Programm enthält lediglich zwei interessante Operationen:

1. Lade Wert x in Register *r16*
2. Speicher Register *r16* in dem SRAM an Adresse *0x0100*

Die Legende gibt Auskunft über den geladenen Wert. Anhang A.1 stellt das zugehörige Programm dar. Das Laden (1) des Wert geschieht in Zeile 22, das Speichern (2) in Zeile 28.

Es sind insgesamt neun Graphen abgebildet. Ein einzelner Graph entspricht hierbei der Messung des Stromverbrauchs bei der Ausführung des Programms zu gegebenem Wert x, mit $x \in \{00000000_2, 00000001_2, 00000011_2, \ldots, 11111111_2\}$. Besonders beachtenswert sind die beiden Zeiträume von 375,0 ns bis 437,5 ns und von 750,0 ns bis 875,0 ns. In dem Zeitraum 375,0 ns bis 437,5 ns wird der *ldi* Befehl ausgeführt und somit der Wert x in das Register *r16* geladen. Zeitraum 750,0 ns bis 875,0 ns stellt den *sts* Befehl dar, hierbei wird das Register *r16* an die Speicheradresse *0x0100* des SRAM geschrieben.

Deutlich zu erkennen ist hierbei der unterschiedliche Stromverbrauch der beiden Operationen. Der *sts* Befehl benötigt mehr Strom als der *ldi* Befehl. Anders ausgedrückt lässt das Speichern eines Registers in den SRAM einen höheren Stromverbrauch verzeichnen, als das Laden eines Wert in ein Register. Die Abhängigkeit des Stroms zur durchgeführten Operation ist ersichtlich.

Aber auch die unterschiedlichen Graphen zeigen ein entscheidendes Merkmal. Je mehr Bits gesetzt sind, desto höher ist der Stromverbrauch. Man kann sagen, dass der Stromverbrauch proportional zur Anzahl der gesetzten Bits ist. Dieses Phänomen beruht auf dem in Abschnitt 4.2.2 beschriebenen Aufbau eines Microcontrollers. Ein hypothetischer Stromverbrauch lässt sich demnach durch das Hamming-Gewicht (siehe Abschnitt 4.4.1), der auf dem Datenbus übertragenden Daten, modellieren.

4.4 Power Models

Bei Power Analysis Attacks ist es oftmals nötig, berechnete Daten auf hypothetische Stromverbrauchswerte abzubilden. Verwendung finden hierbei verschiedene Modelle, die auch als sogenannte Power Models zusammengefasst werden. Im Folgenden wird auf zwei wesentliche Modelle eingegangen[2]. Hierbei handelt es sich um das

[2] Die Beschreibung der beiden Modelle erfolgt anhand binär dargestellter Zahlen

Hamming-Gewicht-Modell und das Hamming-Distanz-Modell.

4.4.1 Hamming-Gewicht-Modell

Das *Hamming-Gewicht* beschreibt den Abstand eines Datenworts zu einem Nullwort. Um das Hamming-Gewicht eines Datum anzugeben, muss man dieses in seiner Binärdarstellung betrachten. Das Nullwort entspricht der Zahl 0. Hierdurch lässt sich das Hamming-Gewicht eines Datum durch die Anzahl der von 0 verschiedenen Bits angeben.

Beispiele: $HW(00000000_2) = 0, \quad HW(11111111_2) = 8, \quad HW(00111001_2) = 4$

Hierbei entspricht $HW(\cdot)$ dem Hamming-Gewicht (engl. *Hamming weight*). Durch das Hamming-Gewicht wird somit angenommen, dass eine unmittelbare Abhängigkeit zwischen den zu verarbeitenden Daten und dem Stromverbrauch vorliegt.

4.4.2 Hamming-Distanz-Modell

Die *Hamming-Distanz* gibt ebenso wie das Hamming-Gewicht einen Abstand an. Es handelt sich hierbei jedoch um den Abstand zwischen einem Datenwort w und einem Referenzwort r. Unter Betrachtung von w und r als binäre Zeichenfolge, gibt die Hamming-Distanz die Anzahl der $0 \rightarrow 1$ und $1 \rightarrow 0$ Übergänge an. Vereinfacht ausgedrückt lässt sich der Vergleich durch eine XOR-Operation realisieren, sodass sich die Hamming-Distanz ebenfalls als $HD(w, r) = HW(w \oplus r)$ ausdrücken lässt. $HD(\cdot)$ steht für die Hamming-Distanz.

Beispiele: $HD(00000000_2, 11111111_2) = 8, \quad HD(01100100_2, 11100011_2) = 4$

Bei der Hamming-Distanz wird nicht von einer alleinigen Abhängigkeit zwischen den zu verarbeitenden Daten und dem Stromverbrauch ausgegangen. Der hypothetische Stromverbrauch bestimmt sich vielmehr durch die Unterschiede zweier Zustände.

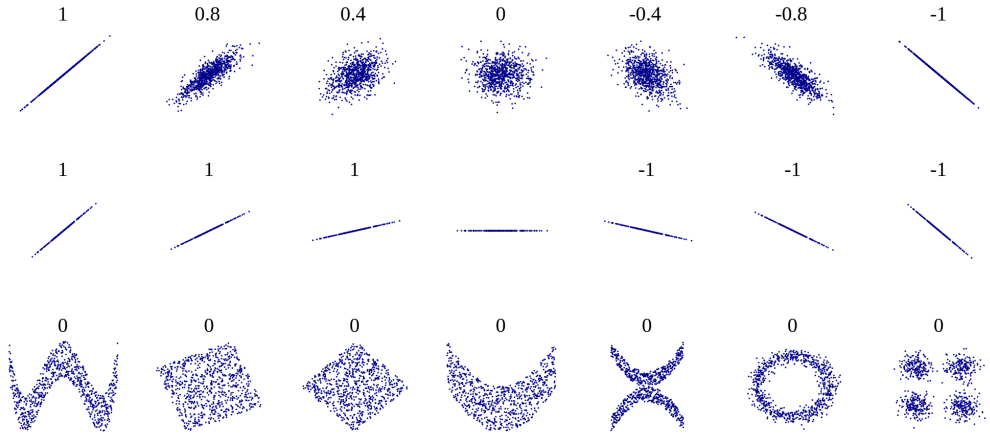
4.5 Bravais-Pearson-Korrelation

Das DPA Verfahren der Power Analysis Attacks beruht auf der Beziehung zweier Merkmale. Dafür wird ein Korrelationsverfahren benötigt. Die Bravais-Pearson-Korrelation, oft auch Pearson-Korrelation oder einfach nur Korrelationskoeffizient genannt, gibt ein Maß für den Zusammenhang zweier Merkmale an. Dieses Maß kann bei einer DPA verwendet werden. Es handelt sich hierbei um ein dimensionsloses Maß mit einem Wertebereich von -1 bis +1. Bei einem Wert von +1 spricht man von einem vollständig positiven linearen Zusammenhang, bei einem Wert von -1 hingegen

von einem vollständig negativen linearen Zusammenhang. Dagegen besteht bei einem Korrelationskoeffizienten von 0 kein linearer Zusammenhang zwischen den beiden Merkmalen. Gleichung 4.1 zeigt die zugrunde liegende Formel für die Berechnung des Korrelationskoeffizienten r für die Messreihe $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ der beiden Merkmale x und y. $\bar{x}$ und $\bar{y}$ entsprechen hierbei dem arithmetischen Mittel des jeweiligen Merkmals.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.1}$$

In Abbildung 4.5 sind eine Reihe an Punkten (x, y) mit den dazugehörigen Korrelationskoeffizienten gegeben. Es ist zu erkennen, dass der Korrelationskoeffizient r



Bildquelle: [Comb]

Abbildung 4.5: Einige Beispiele für die Bravais-Pearson-Korrelation.

sowohl über die Streuung der einzelnen Punkte als auch über die Richtung der linearen Abhängigkeit Angaben macht. Diese beiden Eigenschaften sind in der ersten Zeile der Abbildung erkennbar. Umso höher die Streuung der Punkte, desto mehr konvergiert der Korrelationskoeffizient gegen 0. Dies bedeutet, dass die beiden Merkmale somit nicht korrelieren. Bei einem vollständig positiven oder negativen linearen Zusammenhang ($r = \pm 1$) liegen alle Punkte auf einer Geraden. Durch das Vorzeichen wird die Richtung der linearen Abhängigkeit bestimmt. Der Korrelationskoeffizient sagt jedoch nichts über die Steigung aus, dies ist in der mittleren Zeile der Abbildung zu erkennen. Es ist zu erkennen, dass die Punkte trotz des Korrelationskoeffizienten von ± 1 eine unterschiedliche Steigung besitzen. Es wird demnach nur angegeben, dass zwei Merkmale korrelieren, jedoch nicht wie stark diese Korrelation ist. Nichtlineare Abhängigkeiten, wie sie in der unteren Zeile zu sehen sind, können durch den Korrelationskoeffizienten nicht registriert werden.

4.6 Verfahren

Power Analysis Attacks lassen sich in zwei unterschiedliche Verfahren kategorisieren. Zum einen gibt es die SPA, zum anderen die DPA. Beide Verfahren beruhen auf der Analyse der Stromaufnahme. Bei der SPA handelt es sich um eine manuelle Analyse der Messungen, wohingegen es sich bei der DPA um eine automatisierte Analyse der Messungen handelt.

4.6.1 Simple Power Analysis

Die Simple Power Analysis (SPA) beschäftigt sich mit der visuellen Analyse von Messungen. Da es sich um eine visuelle Analyse handelt, führt ein Angreifer hierbei nur einige wenige Messungen durch. Im Optimalfall reicht sogar eine einzige Messung aus, um an relevante Informationen zu gelangen. Für eine erfolgreiche SPA müssen aufgrund dieser Tatsache, jedoch verschiedene Voraussetzungen erfüllt sein.

Da es sich um eine geringe Anzahl an Messungen handelt, sollten diese ziemlich genau sein. Dies bedeutet, dass Störungen wie zum Beispiel Rauschen auf ein Minimum zu reduzieren sind. Zudem muss ein Angreifer relativ genaue Kenntnis über die konkrete Implementierung des Systems verfügen, um relevante Informationen in den Messungen zu identifizieren. Fernerhin müssen die ausgeführten Operationen abhängig von dem geheimen Schlüsselmaterial sein. Ohne diese Voraussetzungen ist eine SPA nicht möglich.

Mit Hilfe der SPA ist zum Beispiel das Extrahieren des Schlüssels einer RSA-Verschlüsselung möglich. Hierbei kann man im günstigsten Fall einzelne Bits des Schlüssels direkt aus der Messung des Stromverbrauchs ablesen. Dies trifft jedoch nur bei bestimmten Implementierungen des RSA Algorithmus zu. So kommt zum Beispiel für die modulare Exponentiation häufig das sogenannte Square & Multiply Verfahren zum Einsatz, welches gegenüber der SPA verwundbar ist [MOV96, S. 613 ff.].

4.6.2 Differential Power Analysis

Sofern die SPA kein zufriedenstellendes Ergebnis liefert, zum Beispiel aufgrund von Messungenauigkeiten oder keiner bestehenden Abhängigkeit zwischen durchgeführter Operation und geheimen Schlüssel, kann man auf die Differential Power Analysis (DPA) zurückgreifen. Bei diesem Verfahren führt ein Angreifer mehrere Messungen des Stromverbrauchs mit unterschiedlichen Daten durch. Zusätzlich werden hypothetische Annahmen über mögliches Schlüsselmaterial berechnet und mit den tatsächlichen Messungen des Stromverbrauchs verglichen. Im Gegensatz zur visuellen Analyse finden hierbei statistische Methoden ihre Anwendung. Jedoch müssen auch bei diesem

Verfahren einige Voraussetzungen erfüllt sein.

Es muss ein Zwischenergebnis des Algorithmus, welches das Kryptosystem ausführt, berechenbar sein. Dieses Zwischenergebnis muss von Daten, die dem Angreifer bekannt sind, abhängen. Bei den Daten handelt es sich in der Regel um den Klartext oder das Chiffrat. Des Weiteren muss der Stromverbrauch abhängig von den Eingabedaten sein.

Durch die DPA kann ein Angreifer zum Beispiel an den verwendeten Schlüssel einer AES-Verschlüsselung gelangen. Es existiert eine allgemeine Vorgehensweise für diese Art von Angriffen. Dieses Vorgehen lässt sich in fünf auf sich aufbauenden Schritten unterteilen [MOP07, S. 120 ff.], welche wie folgt zusammengesetzt sind:

Schritt 1: Auswahl eines Zwischenergebnisses des ausgeführten Algorithmus

Im ersten Schritt muss ein Zwischenzustand als eine Funktion $f(d, k)$ festgelegt werden. Diese Funktion beschreibt ein Zwischenergebnis des Algorithmus, welcher auf dem anzugreifenden kryptographischen System ausgeführt wird. d entspricht hierbei einem bekannten, nicht konstanten Datum, wohingegen k einen Teil des verwendeten Schlüsselmaterials darstellt. Im Allgemeinen handelt es sich bei dem Datum d entweder um den Klartext oder das Chiffrat eines Verschlüsselungsverfahrens.

Schritt 2: Messen des Stromverbrauchs

Als nächstes geht es um die tatsächliche Messung des Stromverbrauchs des kryptographischen Systems. Hierfür wird der Stromverbrauch gemessen, währenddessen D verschiedene Datenblöcke von dem anzugreifenden kryptographischen System verschlüsselt oder entschlüsselt werden. Für alle D Durchläufe muss das Datum d bekannt sein, da dieses für die unter Schritt 1 festgelegte Funktion benötigt wird. Die Datenblöcke können als Vektor $\boldsymbol{d} = (d_1, \ldots, d_D)'$ aufgefasst werden, wobei d_i dem Datum des i-ten Durchlaufs der Verschlüsselung oder Entschlüsselung entspricht. Da zu jedem einzelnen Durchlauf eine Messung des Stromverbrauchs durchgeführt wird, existiert zu jeglichem Datenblock d_i, ein Vektor $\boldsymbol{t_i'} = (t_{i,1}, \ldots, t_{i,T})$ mit T Messpunkten. Die aufgezeichneten Traces können somit als eine Matrix $\boldsymbol{T}$ der Größe $D \times T$ aufgefasst werden.

Schritt 3: Berechnung hypothetischer Zwischenwerte

Der nächste Schritt beschäftigt sich mit der Berechnung von hypothetischen Zwischenwerten für jede mögliche Schlüsselhypothese k. Die unterschiedlichen Schlüsselhypothesen können als Vektor $\boldsymbol{k} = (k_1, \ldots, k_K)$ mit K verschiedenen Möglichkeiten aufgefasst werden. Mit Hilfe von $\boldsymbol{d}$, $\boldsymbol{k}$ und der unter Schritt 1 festgelegten Funktion $f(d, k)$ können somit für alle D Datenblöcke und alle

K Schlüsselhypothesen hypothetische Zwischenwerte berechnet werden. Das Ergebnis kann durch eine Matrix $\boldsymbol{V}$ der Größe $D \times K$ mit

$$v_{i,j} = f(d_i, k_j) \quad i = 1, \ldots, D \quad j = 1, \ldots, K$$

beschrieben werden. Die Spalten j der Matrix $\boldsymbol{V}$ beinhalten somit die berechneten Zwischenwerte unter der Annahme des Schlüssels k_j. Da $\boldsymbol{k}$ alle möglichen Schlüsselhypothesen enthält, muss eine dieser Spalten die Zwischenwerte enthalten, welche ebenfalls auf dem anzugreifenden System während der D Durchläufe berechnet wurden. Das Ziel der DPA ist es, herauszufinden, welche Spalte von $\boldsymbol{V}$ während der D Durchläufe tatsächlich von dem anzugreifenden kryptographischen System berechnet wurde.

Schritt 4: Mapping der hypothetischen Zwischenwerte auf hypothetische Stromverbrauchswerte

Nachdem die hypothetischen Zwischenwerte berechnet wurden, müssen diese auf entsprechende hypothetische Stromverbrauchswerte abgebildet werden. Dies ist nötig, um einen späteren Vergleich mit den tatsächlichen Traces durchführen zu können. Bei dieser Operation wird jedes $v_{i,j} \in \boldsymbol{V}$ durch ein entsprechendes Modell auf ein Element $h_{i,j}$ gemappt. Die daraus resultierende Matrix wird mit $\boldsymbol{H}$ bezeichnet. Als Modell wird ein sogenanntes Power Model verwendet (siehe Abschnitt 4.4). Dieses stellt eine Simulation des Stromverbrauchs der hypothetisch berechneten Zwischenwerte für unterschiedliche Schlüsselhypothesen dar und ist maßgeblich für den Erfolg oder Misserfolg der DPA verantwortlich.

Schritt 5: Vergleich der hypothetischen Stromverbrauchswerte mit den aufgezeichneten Traces

Im letzten Schritt findet ein Vergleich der hypothetischen Stromverbrauchswerte mit den aufgezeichneten Traces statt. Hierbei wird jede Spalte $\boldsymbol{h_i}$ der Matrix $\boldsymbol{H}$ mit jeder Spalte $\boldsymbol{t_j}$ der Matrix $\boldsymbol{T}$ verglichen. Es werden somit die hypothetischen Stromverbrauchswerte jeder Schlüsselhypothese mit jedem Zeitpunkt der tatsächlich aufgezeichneten Traces verglichen. Das Ergebnis dieser Operation kann als Matrix $\boldsymbol{R}$ der Größe $K \times T$ dargestellt werden. Jedes Element $r_{i,j}$ entspricht hierbei dem Vergleich der beiden Spalten $\boldsymbol{h_i}$ und $\boldsymbol{t_j}$. Der Vergleich kann durch unterschiedliche Algorithmen erfolgen. Ein gängiges Verfahren ist die Bravais-Pearson-Korrelation, welche in Abschnitt 4.5 näher beschrieben ist.

Das obige aufgeführte Verfahren wird oftmals mit dem Namen Correlation Power Analysis (CPA) bezeichnet. Die Literatur ist sich hierbei jedoch nicht einig. In dieser Arbeit fällt das beschriebene Verfahren unter den Begriff der DPA. Der Grund für

die Unstimmigkeit liegt darin begründet, dass Kocher, Jaffe und Jun, Erfinder der DPA, ursprünglich keine Korrelationsmethode verwendet haben [Koc96].

4.7 Gegenmaßnahmen

Es gibt einige Gegenmaßnahmen, um Power Analysis Attacks zu verhindern beziehungsweise einem Angreifer einen erfolgreichen Angriff zu erschweren. Da ein System einen solchen Angriff jedoch nicht selbst erkennen kann, müssen diese Gegenmaßnahmen bereits während des Entwicklungsprozess des kryptographischen Systems mitbedacht werden.

Angriffe per SPA können relativ leicht verhindert werden. Hierbei sind oftmals bedingte Sprünge für die Angreifbarkeit eines kryptographischen Systems verantwortlich. Indem darauf geachtet wird, keine bedingten Sprünge in Abhängigkeit des geheimen Schlüsselmaterials auszuführen, kann die Angreifbarkeit vermieden werden. Ebenso kann das Einfügen von Rauschen eine erfolgreiche SPA verhindern.

Für die DPA existieren verschiedene Gegenmaßnahmen. So kann auch hier das Einfügen von Rauschen eine sinnvolle Gegenmaßnahme sein. Durch das Einfügen von Zufallsoperationen kann die Angreifbarkeit nochmals erschwert werden. Hierdurch kann eine Desynchronisation bei der Ausführung der kryptographischen Operation erreicht werden. Zudem stellt auch die Dual-Rail Precharge Logic [MOP07, S. 177 f.] eine interessante Gegenmaßnahme dar. Durch diese Methode wird versucht, den Stromverbrauch zu jedem Takt möglichst konstant zu halten. Dies kann erreicht werden, indem zu jeder Operation eine komplementäre Dummyoperation ausgeführt wird.

Wie bereits erwähnt, verhindern solche Gegenmaßnahmen einen erfolgreichen Angriff nicht völlig, sondern erschweren diesen oftmals nur. Für viele Gegenmaßnahmen existieren bereits Lösungsansätze, um diese zu umgehen. Eine Desynchronistation kann zum Beispiel durch das Elastic Alignment [WWB11] außer Kraft gesetzt werden.

5 Aufbau einer Laborumgebung

Dieser Abschnitt beschreibt den Aufbau der Laborumgebung. Es wird insbesondere auf die verwendete Hardware und entwickelte Software eingegangen. Zudem wird ein Überblick über mögliche anfallende Kosten gegeben. In Abschnitt 6 wird auf die Durchführung von Power Analysis Attacks mit der hier beschriebenen Laborumgebung eingegangen.

5.1 Verwendete Hardware

Für die Durchführung der Power Analysis Attacks werden einige Hardwarekomponenten verwendet. Tabelle 5.1 gibt einen Überblick über die in der Laborumgebung eingesetzte Hardware. Die Verwendung einer zusätzlichen Grafikkarte ist hierbei

Bezeichnung	Hersteller / Typ
Digitaloszilloskop	Rigol DS1102E
USB In-Circuit-Programmer	USBasp
Regelbares Netzgerät	QJE PS3003
USB2UART Bridge	Baite B75937
(Grafikkarte	Nvidia Geforce GTX 970)

Tabelle 5.1: Überblick über die in der Laborumgebung eingesetzte Hardware.

nur optional (siehe Abschnitt 5.2.2). Im Folgenden wird das verwendete Oszilloskop näher erläutert. Des Weiteren wird auf relevante Kenngrößen dieser Komponenten eingegangen und die weiteren Hardwarekomponenten erläutert.

5.1.1 Digitaloszilloskop

Als Digitaloszilloskop wird das Rigol DS1102E genutzt. Dieses besitzt zwei Kanäle, eine Abtastrate von 1 GSa/s und eine Bandbreite von 100 MHz.

Es verfügt über zwei verschiedene Speichermodi, wodurch die Speichertiefe festgelegt werden kann. Hierbei kann zwischen dem Modus *Normal Memory* und dem Modus *Long Memory* gewählt werden. In dem Normal Memory Modus werden 128^2

Messpunkte gespeichert, im Long Memory Modus hingegen 1024^2 Messpunkte. Diese Angaben beziehen sich auf die Nutzung von nur einem Kanal. Werden zwei Kanäle verwendet, so teilt sich der Speicher für die Messpunkte auf beide Kanäle gleichmäßig auf.

Das Rigol DS1102E bietet verschiedene Triggerquellen als Auslöser an. Ohne einen Trigger kann kein stehendes Signalbild erhalten werden. Somit wäre die Durchführung einer Messung nicht möglich. So kann der Trigger zum Beispiel durch ein Signal an einen der zur Verfügung stehenden Kanäle oder einem externen Triggereingang ausgelöst werden. Durch die zahlreichen Triggermodi steht eine breite Anzahl an zur Verfügung stehenden Ereignissen zur Auswahl. Es kann unter anderem auf eine Flanke oder die Abfall- beziehungsweise der Anstiegsgeschwindigkeit eines Signals getriggert werden. Durch die zahlreichen Triggermöglichkeiten kann somit eine Messung zu einem festgelegten Zeitpunkt erfolgen.

Zudem verfügt das ausgewählte Oszilloskop über eine gut dokumentierte Universal Serial Bus (USB)-Schnittstelle. Über diese Schnittstelle kann das Oszilloskop konfiguriert, Messungen erstellt und Daten empfangen werden. Die Kommunikation geschieht über einfache Kommandos, welche American Standard Code for Information Interchange (ASCII) kodiert übertragen werden. Durch das Kommando `:WAV:DATA? CH1` kann zum Beispiel der Speicher des ersten Kanals ausgelesen werden. Eine solche Schnittstelle ist wichtig, um automatisiert Messungen durchführen zu können.

Auch bei dem Oszilloskop gibt es einige technische Daten, welche man bei der Auswahl beachten sollte. Sowohl die Bandbreite als auch die Abtastrate stellen wichtige Größen dar. Die Bandbreite sollte nicht geringer sein als die Taktfrequenz eines Systems, auf welchem Messungen durchgeführt werden. In dieser Arbeit beruht der Testaufbau auf einem Microcontroller, welcher eine maximale Frequenz von 20 MHz unterstützt. Damit ist das Rigol Oszilloskop ausreichend. Zudem sollte man die Speichertiefe, welche im Zusammenhang mit der Abtastrate steht, nicht vernachlässigen. Es sollte hierbei, bezogen auf das Anwendungsgebiet, auf eine hinreichend genaue Auflösung geachtet werden, sodass alle nötigen Daten erfasst werden können. Sofern nur ein kleiner Zeitraum erfasst werden muss, reichen einige wenige Messpunkte aus, andernfalls werden mehr Messpunkte benötigt.

5.1.2 Weitere Hardware

Es werden noch einige weitere Komponenten verwendet. Da es hierbei nur wenige Details zu beachten gibt, folgt nun eine kurze Beschreibung von diesen Komponenten.

Da in dem späteren Testaufbau (siehe Abschnitt 6) ein Microcontroller der Atmel AVR Familie verwendet wird, kann für das Flashen des ATmega328P der USB In-Circuit-Programmer USBasp verwendet werden. Bei diesem handelt es sich um einen

speziell für Microcontroller der Atmel AVR Familie entwickeltes Programmiergerät. Der USBasp besteht im wesentlichen aus einem Microcontroller und einigen wenigen weiteren Bauteilen. Bis auf den USBasp wird keine weitere Hardware für das Flashen des verwendeten Microcontrollers benötigt.

Zur Energieversorgung des Microcontrollers reicht ein handelsübliches regelbares Netzgerät aus. Gebrauch gemacht wurde hierbei von dem QJE PS3003. Die mögliche Ausgangsspannung von diesem Netzgerät kann man stufenlos von 0 V bis 30 V regeln. Da der Microcontroller mit einer Spannung von 1,8 V bis 5,5 V betrieben werden kann, reicht das PS3003 völlig aus. Es sollte lediglich darauf geachtet werden, dass das Netzteil möglichst kein Rauschen verursacht.

Aufgrund dessen, dass der Microcontroller des Testaufbaus von dem Computer aus angesteuert werden soll, kam hierfür die serielle Schnittstelle in Frage. Dafür wird eine USB2UART Bridge verwendet. Mit dieser kann zwischen dem Microcontroller und dem Computer kommuniziert werden.

5.2 Eigene Software

Für die Verwendung der Laborumgebung wurden einige Programme entwickelt. All diese Programme, die Computerseitig Anwendung fanden, wurden mit Hilfe von *python* erstellt.

5.2.1 Automatisierte Steuerung des Oszilloskop

Die Kommunikation zwischen dem Computer und dem Oszilloskop verläuft über die USB-Schnittstelle. Das verwendete Oszilloskop bietet hierfür eine entsprechende Schnittstelle an, worüber Kommandos direkt als ASCII-kodierte Zeichenketten übertragen werden können. Hiermit kann das Oszilloskop unter anderem konfiguriert werden. Zudem können Messungen durchgeführt und die Daten über die USB-Schnittstelle übertragen werden. Das heißt, durch die ASCII-basierte Schnittstelle kann das Oszilloskop vollständig gesteuert werden. Somit kann die Kommunikation auch auf automatisiertem Wege stattfinden. Es können zum Beispiel die folgenden Kommandos verwendet werden:

`:RUN`
: Durch das `:RUN` Kommando kann man das Oszilloskop in den Modus zum Erfassen von Signaldaten versetzen.

`:STOP`
: Hiermit kann man das Erfassen von Signaldaten durch das Oszilloskop stoppen.

`:WAV:DATA? CHAN1`

Mit Hilfe von diesem Kommando kann man den Datenspeicher des Oszilloskops auslesen. Dieses Beispiel liest die gesamten Daten des ersten Kanals (`CHAN1`) aus.

`:WAV:POIN:MODE RAW`

Das Oszilloskop verfügt über verschiedene Speichermodi, die man durch dieses Kommando setzen kann. Um den vollständigen Speicher des Oszilloskops auszulesen, muss man diesen vorerst in den `RAW` Modus versetzen.

`:TRIG:STAT?`

Durch dieses Kommando kann man den aktuellen Status des Oszilloskops abfragen, um beispielsweise das Auslösen des Triggers zu überprüfen und darauf zu reagieren. Dieser kann unter anderem die Zustände `RUN` und `STOP` annehmen. Wie bereits weiter oben erwähnt, erfasst das Oszilloskop im Zustand `RUN` die Signaldaten kontinuierlich. Hingegen findet im Zustand `STOP` keine Erfassung von Signaldaten statt.

Eine Übersicht über alle verfügbaren Kommandos kann man der Schnittstellenbeschreibung des Herstellers entnehmen. Diese ist unter [RIG13] einzusehen.

5.2.2 Optimierung mittels CUDA

Die in Abschnitt 4.5 beschriebene Pearson-Korrelation wird während einer DPA ausgeführt. Der Korrelationskoeffizient stellt im Allgemeinen keine komplexe Operation dar, muss jedoch vielfach ausgeführt werden.

Da das Berechnen des Korrelationskoeffizienten auf Operationen auf Vektoren beruht, die parallel ausgeführt werden können, liegt der Einsatz einer Grafikkarte nahe. Durch den Einsatz einer Graphics Processing Unit (GPU) kann zusätzlich zur CPU weitere Rechenkapazität zur Verfügung gestellt werden. GPUs zeichnen sich insbesondere durch ein hohes Maß an Parallelisierung aus. Verwendet wurde hierfür eine Nvidia Geforce GTX 970. Diese unterstützt die Compute Unified Device Architecture (CUDA)-Technologie mit Hilfe derer die Abarbeitung von Programmcode durch die GPU ermöglicht wird. Die Geforce GTX 970 besitzt 1664 CUDA-Recheneinheiten, sodass hoch parallelisiert gerechnet werden kann.

Es gibt bereits ein entsprechendes Python Modul *PyCUDA*, durch welches die CUDA-Schnittstelle aus Python heraus angesprochen werden kann. Hierdurch wird eine einfache Programmierung ermöglicht. Dieses Modul wurde verwendet, um die Berechnung des Korrelationskoeffizienten auf die GPU auszulagern.

5.2.3 Software für Graphische Ausgaben

Power Analysis Attacks beruhen auf Messergebnissen mit einer Vielzahl an Messdaten. Diese Daten lassen sich sinnvoll als Grafik aufbereiten. Zum einen kann hierdurch die Überprüfung der Daten auf Korrektheit durchgeführt werden, zum anderen ist es oftmals verständlicher bestimmte Merkmale direkt von einer Grafik abzulesen.

Alle erstellten Programme für die Graphische Darstellung der Messdaten wurden mit Hilfe von *python* erstellt. Hierbei fand das Python Modul *Matplotlib* Anwendung. Bei Matplotlib handelt es sich um eine Bibliothek, mit der Daten verschiedener Art als Graphen dargestellt werden können. Die Bibliothek ist dabei angelehnt an Matlab, eine Software zur Lösung und Darstellung mathematischer Probleme.

In dieser Arbeit beinhalten einige Abbildungen Grafiken mit Messwerten. All diese Grafiken wurden mit Hilfe der selbst erstellten Programme erstellt.

5.3 Kosten

Abschnitt 5.1 zeigt die eingesetzte Hardware. Es ist ersichtlich, dass keine besonderen Hardwarekomponenten nötig sind, um eine Laborumgebung für Power Analysis Attacks aufzubauen. Somit verursacht der Aufbau einer Laborumgebung keine hohen Kosten. Tabelle 5.2 stellt die Kosten tabellarisch dar. Die einzigen Kosten, die bedacht

Bezeichnung	Preis (€)
Digitaloszilloskop	400
USB In-Circuit-Programmer	3
Regelbares Netzgerät	50
USB2UART Bridge	5
Grafikkarte	300
	758

Tabelle 5.2: Überblick über die Kosten für die in der Laborumgebung eingesetzte Hardware.

werden sollten, sind die Kosten für ein Digitaloszilloskop und optional für eine Grafikkarte, um die Rechenkapazität zu erhöhen. Die restlichen Hardwarekomponenten sind von den Kosten her nicht erwähnenswert. Zum Zeitpunkt dieser Arbeit waren das Oszilloskop für 400 € und die Grafikkarte für knapp 300 € zu erwerben. Die Gesamtkosten belaufen sich auf 758 €.

6 Überprüfung der Laborumgebung anhand eines Testaufbaus

Die Überprüfung der in Abschnitt 5 beschriebenen Laborumgebung fand anhand eines Testaufbaus unter Laborbedingungen statt. Die nachfolgenden Abschnitte gehen auf das anzugreifende kryptographische System, die benötigte Software, die elektronische Schaltung und die Kommunikation zwischen Computer und Microcontroller ein. Zudem wird die Durchführung der SPA und DPA beschrieben.

6.1 Anzugreifendes System

Alle durchgeführten Versuche wurden auf einem ATmega328P vorgenommen. Hierbei handelt es sich um einen 8-Bit Microcontroller des Herstellers Atmel. Dieser Microcontroller gehört der Atmel AVR Familie an. Die durchgeführten Angriffe sollten problemlos auf die Microcontroller ATmega48A/PA, ATmega88A/PA, ATmega168A/PA und ATmega328 übertragbar sein. Diese unterscheiden sich gegenüber dem ATmega328P lediglich geringfügig. Die Unterschiede umfassen zum Beispiel die Größe des Speichers oder die Größe des Interruptvektors. Alle Daten wurden dem offiziellen Datenblatt [Atm15] entnommen.

Die gültige Versorgungsspannung liegt bei diesen Modellen in dem Bereich von 1,8 V bis 5,5 V. Eine Verbindung mit dem Microcontroller kann über die frei programmierbaren I/O Pins hergestellt werden. Hierfür stehen 23 Pins zur Verfügung.

Der ATmega328P beruht auf der Reduced Instruction Set Computer (RISC) Architektur, sodass die meisten Befehle innerhalb von einem Systemtakt abgearbeitet werden. Insgesamt bietet der ATmega328P einen Befehlssatz mit 131 verschiedenen Befehlen. Ferner sind Taktfrequenzen von bis zu 20 MHz möglich, sofern die Versorgungsspannung zwischen 4,5 V bis 5,5 V liegt. Eine Besonderheit sind die „voll statischen Operationen“ (engl. *fully static operation*), wodurch der ATmega328P bis auf 0 Hz runtergetaktet werden kann, ohne Daten zu verlieren.

Um Parallel sowohl auf den Programmspeicher als auch den Datenspeicher zugreifen zu können, entspricht der Aufbau des ATmega328P der Harvard-Architektur. Ebenfalls ist hierdurch eine hohe Performance gewährleistet. Durch diese Architektur

existieren somit getrennte Adressräume für die einzelnen Speichertypen. Der ATmega328P besitzt insgesamt drei verschiedene Speichertypen: Den Flash-Speicher, den EEPROM und den SRAM. Der Flash-Speicher dient als Programmspeicher. In diesem wird das eigentliche Programm abgelegt. Der EEPROM ist ein nichtflüchtiger Speicher und kann für Daten verwendet werden, welche auch nach dem Ausschalten des Microcontrollers noch zur Verfügung stehen sollen. Hingegen stellt der SRAM einen flüchtigen Speicher dar, sodass gespeicherte Informationen bei dem Abschalten des Microcontrollers verloren gehen.

Zwei wichtige Kenngrößen bei der Auswahl des Microcontrollers sind unter anderem die Größe des Flash-Speichers und die Größe des Arbeitsspeichers (SRAM). Der ATmega328P besitzt einen 32 KB großen Flash-Speicher und 2 KB Arbeitsspeicher. Die Auswahl richtete sich hierbei nach der Größe der entwickelten Testprogramme. Es musste insbesondere darauf geachtet werden, dass genug Speicher für eine AES Implementierung zur Verfügung stand. Für alle durchgeführten Versuche sind die genannten Speichergrößen ausreichend gewesen.

6.2 Verwendete Software

Für die Durchführung der Power Analysis Attacks werden einige fertige Softwarekomponenten eingesetzt. Tabelle 6.1 gibt einen Überblick über die in dem Testaufbau eingesetzte Software, mitsamt ihren jeweiligen Versionen.

Softwarepaket	Version
avr-gcc	4.8.1
avr-binutils	2.24
avr-libc	1.8.0
avrdude	6.1

Tabelle 6.1: Überblick über die in dem Testaufbau eingesetzte Software.

Um ein Programm für Microcontroller der AVR Familie zu erstellen, kann die AVR-Toolchain verwendet werden. Diese besteht im Kern aus dem *avr-gcc*, den *avr-binutils* und der *avr-libc*. Für das Übertragen des Programms auf den Microcontroller kann *avrdude* genutzt werden.

6.2.1 AVR-Toolchain (avr-gcc, avr-binutils, avr-libc)

Durch die AVR-Toolchain stehen unterschiedliche Kommandozeilenprogramme zur Verfügung. Bestandteile der Toolchain sind insbesondere der avr-gcc, die avr-binutils

und die avr-libc.

Der avr-gcc ist ein freier Compiler. Mit Hilfe des avr-gcc können in C geschriebene Programme für Microcontroller der AVR Familie übersetzt werden. Das Übersetzen geschieht in Zusammenarbeit mit weiteren Hilfsprogrammen wie dem Präprozessor, dem Assembler und dem Linker. Das Resultat des Compilieren ist eine Binärdatei.

Die avr-binutils enthalten zahlreiche Hilfsprogramme für die Manipulation von Dateien für AVR-Microcontroller. So sind darin sowohl der Assembler *avr-as* als auch der Linker *avr-ld* enthalten. Zudem ist auch das Tool *avr-objcopy* enthalten. Hiermit ist es möglich, diverse Dateiformate in andere Formate zu übersetzen. Dies wird benötigt, um Dateien im Intel HEX-Format zu erzeugen.

Durch die avr-libc steht eine C-Standard-Bibliothek zur Verfügung. Genau genommen handelt es sich um eine Teilmenge der C-Standard-Bibliothek, da im Embedded Bereich einige Einschränkungen vorgenommen werden müssen. Zusätzlich enthalten sind darin AVR-spezifische Header-Dateien und Funktionen. Die avr-libc erleichert den Zugriff auf die Hardware erheblich.

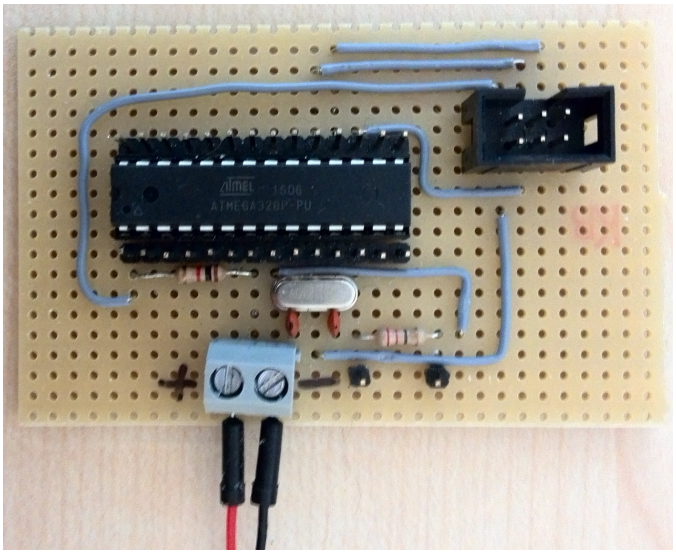
6.2.2 avrdude

Avrdude ist eine kommandozeilenbasierte Programmiersoftware für Atmel AVR-Microcontroller. Durch avrdude kann man Inhalte der Speicherbereiche (z.B. des EEPROM) eines AVR-Microcontrollers beschreiben, auslesen oder verändern. Somit kann mit Hilfe von avrdude zum Beispiel Programmcode in den Flash-Speicher eines Microcontrollers übertragen werden. Zudem ist auch das Setzen und Lesen der Fuse-Bits möglich. Mittels der Fuse-Bits ist beispielsweise das Einstellen des internen Takts machbar.

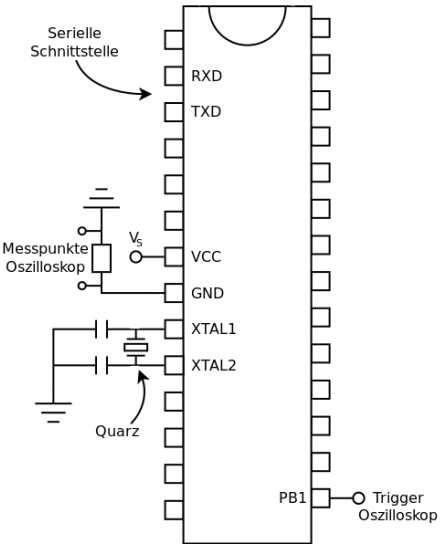
6.3 Elektronische Schaltung

Damit eine flexible Handhabung des Microcontrollers gewährleistet ist, wird eine eigens entwickelte Schaltung verwendet. Wie in Abbildung 6.1 zu sehen, kann diese elektronische Schaltung auf einer Lochrasterplatine aufgebaut werden.

Die Schaltung enthält lediglich eine geringe Anzahl an Bauteilen. So wird zusätzlich zu dem Microcontroller ein externer Quarz mit einer Frequenz von 16 MHz als Taktgeber verbaut. Der Microcontroller befindet sich auf einem Sockel, sodass ein Austausch von diesem problemlos möglich ist. Zusätzlich ist ein 6-poliger Wannenstecker zum einfachen Anschließen des Programmiergeräts vorhanden. Alle Pins des Microcontrollers sind durch entsprechende Stiftleisten ausgeführt, um das Herstellen einer Verbindung zu ermöglichen. Um das Messen des Stromverbrauchs zu vereinfachen,



(a) Elektronische Schaltung.



(b) Schematische Darstellung der Schaltung.

Abbildung 6.1: In der linken Grafik ist die eingesetzte elektronische Schaltung des Testaufbaus zu sehen. Die rechte Grafik enthält eine schematische Darstellung der Schaltung und wichtiger verwendeter Pins.

ist der Widerstand über den gemessen wird, bereits fest in der Schaltung verbaut. Hierbei handelt es sich um einen 22 Ω Widerstand. Ebenfalls ist eine schematische Darstellung der Schaltung zu sehen. Über die Pins *RXD* und *TXD* findet die serielle Kommunikation statt. Näheres dazu wird in dem nachfolgenden Abschnitt beschrieben. Durch *PB1* wird ein entsprechendes Signal für das Oszilloskop erzeugt, sodass dieser triggern kann. Zudem sind der Widerstand und der Quarz in der schematischen Darstellung zu erkennen.

6.4 Kommunikation zwischen Computer und Microcontroller

Bei der Kommunikation zwischen dem Computer und dem Microcontroller wird auf die serielle Schnittstelle gesetzt. Der Austausch von Daten zwischen diesen beiden Komponenten beschränkt sich auf das Übermitteln der Klartexte und den Chiffraten. Hierbei werden immer Blöcke mit einer Größe von 16 Bytes übermittelt. Seitens des Computers wird die serielle Schnittstelle über ein Python Skript angesteuert. Hierfür kommt das Modul *pySerial* zum Einsatz. Damit kann man einfach auf die serielle Schnittstelle zugreifen und über diese kommunizieren. Listing 6.1 zeigt ein Beispiel für die Initialisierung der Schnittstelle (Zeile 3-8).

```
1  import serial
2
3  ser = serial.Serial(
4      port='/dev/ttyUSB0',
5      baudrate=9600,
6      parity=serial.PARITY_NONE,
7      stopbits=serial.STOPBITS_TWO,
8      bytesize=serial.EIGHTBITS)
9
10 plaintext = '0123456789abcdef'
11 for i in range(len(plaintext)): ser.write(bytes([ord(plaintext[i])]))
12
13 recv_ct = ser.read(len(plaintext))
```

Listing 6.1: Beispiel für die Initialisierung und den Gebrauch der seriellen Schnittstelle des Microcontrollers, unter Verwendung des Python Moduls pySerial.

Wichtige Konfigurationsparameter sind hierbei die Baudrate und das Übertragungsformat. Als Geschwindigkeit werden 9600 Baud gewählt. Das Übertragungsformat entspricht 8N2. Demnach folgen auf ein Startbit insgesamt acht Datenbits, kein Paritätsbit und zwei Stoppbits. Ebenfalls in Listing 6.1 zu erkennen sind sowohl ein Schreibvorgang (Zeile 11) als auch ein Lesevorgang (Zeile 13).

6.5 Simple Power Analysis

Durch diesen Abschnitt wird gezeigt, wie ein Angreifer mit Hilfe der SPA an das Schlüsselmaterial einer RSA-Verschlüsselung gelangen kann. Konkret wird auf die binäre Exponentiation eingegangen, welche ebenfalls als Square-and-Multiply Verfahren bekannt ist.

Die binäre Exponentiation beschäftigt sich mit der Berechnung von natürlichen Potenzen. Mit ihr können Ausdrücke der Form x^k mit $k \in \mathbb{N}$ effizient berechnet werden. Unter Effizienz versteht man hierbei das Einsparen von Multiplikationen. So lässt sich die Potenz $y = x^k$ mit $k = 19$ zum Beispiel durch $y = \underbrace{x \cdot \ldots \cdot x}_{k-mal}$ oder durch $y = ((((x^2)^2)^2) \cdot x)^2 \cdot x$ darstellen. Letzteres entspricht hierbei dem Prinzip der binären Exponentiation, y lässt sich somit ausschließlich durch mehrmaliges quadrieren und multiplizieren mit x berechnen. Der Algorithmus für eine solche Darstellung lässt sich, wie von Knuth [Knu81, S. 441] vorgestellt, folgendermaßen beschreiben:

- Wandle k zunächst in die zugehörige Binärdarstellung, ohne führende Nullen,

um.

- Substituiere daraufhin jede 1 durch die Zeichenkette SM und jede 0 durch das Zeichen S. S steht dabei für das Quadrieren (engl. *to square*) und M für das Multiplizieren.
- Die daraus resultierende Zeichenkette bildet die Regel für die Berechnung von x^k. Hierfür muss man die Zeichenkette von links nach rechts durchlaufen und beginnend mit der Zahl 1, das bisherige Zwischenergebnis für jedes Vorkommen des Zeichen S quadrieren und für jedes Zeichen M mit x multiplizieren.
- *Vereinfachte Variante: Die ersten beiden Zeichen der obigen gebildeten Zeichenkette, somit das erste SM, kann man streichen. Hierdurch kann man bei der Berechnung von x^k direkt mit x, anstelle der Zahl 1, als Zwischenergebnis beginnen, da $1^2 \cdot x = x$ gilt.*

Die Berechnung kann somit in Abhängigkeit der einzelnen Bits des Exponenten durchgeführt werden. Um zur obigen Darstellung $y = x^{19} = ((((x^2)^2)^2) \cdot x)^2 \cdot x$ zu gelangen, muss die 19 somit zuerst in ihre Binärdarstellung zerlegt werden ($19_{10} = 10011_2$). Durch Substitution der Einsen und Nullen erhalten wir die Zeichenkette SMSSSMSM, wobei das erste SM entfällt (Vereinfachte Variante) und somit SSSMSM übrig bleibt. Nun muss beginnend mit x, dreimal quadriert, einmal mit x multipliziert, quadriert und nochmals mit x multipliziert werden, sodass wir $y = ((((x^2)^2)^2) \cdot x)^2 \cdot x$ erhalten.

Es wird gezeigt wie der Exponent k der binären Exponentiation direkt aus dem Stromverbrauch visuell abgelesen werden kann. Die binäre Exponentiation im Allgemeinen stellt natürlich keine Verschlüsselung dar. Dieses Verfahren wird jedoch oftmals zum Beispiel bei der RSA-Verschlüsselung angewendet, wobei der Exponent k dem tatsächlichen RSA-Schlüssel entspricht. Hierdurch kann eine RSA-Implementierung, die dieses Verfahren verwendet, gegenüber der SPA verwundbar sein.

6.5.1 Verwendetes Programm

Um eine SPA auf das Square-and-Multiply Verfahren durchzuführen, wurde der Algorithmus zuerst implementiert. Das in Listing 6.2 gezeigte Programm[1] wurde hierfür erstellt.

```
1 #include <avr/io.h>
2 #include <stdint.h>
```

[1] Die vorliegende Implementierung des Square-and-Multiply Verfahrens entspricht nicht ganz dem beschriebenen Verfahren. In dieser Implementierung wird der Exponent in umgekehrter Reihenfolge durchlaufen, demnach bitweise vom Least Significant Bit (LSB) zum Most Significant Bit (MSB).

```
3  #include <stdlib.h>
4
5  #define trigger_init() DDRB |= 1<<DDB1
6  #define trigger_high() PORTB |= 1<<PORTB1
7  #define trigger_low() PORTB &= ~(1<<PORTB1)
8
9  uint8_t binary_exp(uint8_t b, uint8_t n) {
10     uint8_t res = 1;
11     while (n) {
12         if (n & 1) {
13             res *= b; // multiply
14         }
15         b *= b; // square
16
17         n >>= 1;
18     }
19     return res;
20 }
21
22 uint8_t key = 0b00010011;
23 uint8_t m = 3;
24
25 int main (void) {
26     trigger_init();
27
28     while (1) {
29         trigger_high();
30         trigger_low();
31
32         binary_exp(m, key);
33     }
34
35     return 0;
36 }
```

Listing 6.2: Verwendetes Programm bei der Durchführung der SPA

Die Funktion *binary_exp*(b, n) (Zeile 9-20) beschreibt das Square-and-Multiply Verfahren. Sie erhält zwei Übergabeparameter, b und n. Hierbei entspricht b der Basis und n dem Exponenten, wobei es sich bei dem Exponenten, wie bereits eingangs erwähnt, zum Beispiel um den Schlüssel einer RSA-Verschlüsselung handeln kann. Der Rückgabewert entspricht dem Ergebnis von b^n. Es ist deutlich erkennbar, wie der Algorithmus den Exponenten bitweise durchläuft. In Abhängigkeit des Zustandes jedes einzelnen Bits, folgt entweder eine Quadrierung oder sowohl eine Quadrierung als auch eine Multiplikation.

In dem Hauptprogramm wird die Square-and-Multiply Funktion aufgerufen (Zeile 32). Die übergebenen Parameter sind hierbei in den Zeilen 22 und 23 auf konstante Werte festgelegt. Um die Messungen mit dem Oszilloskop zum richtigen Zeitpunkt durchzuführen, wird in den Zeilen 29 und 30 noch ein entsprechendes Signal zum Triggern erzeugt.

6.5.2 Durchführung

Für die Durchführung der SPA auf das Square-and-Multiply Verfahren wird das in Abschnitt 6.5.1 beschriebene Programm auf dem Microcontroller der Laborumgebung ausgeführt. Mit Hilfe des Oszilloskops kann daraufhin eine Messung des Stromverbrauchs durchgeführt werden. Die vorliegende SPA beschränkt sich auf die visuelle Auswertung von einem Trace. Um Störungen, wie zum Beispiel Rauschen zu vermindern, handelt es sich bei diesem Trace um das berechnete arithmetische Mittel mehrerer durchgeführter Messungen.

In Abbildung 6.2 ist der aufgezeichnete Trace zu dem in Abschnitt 6.5.1 beschriebenen Programm zu sehen. Dieser Trace stellt das arithmetische Mittel von insgesamt

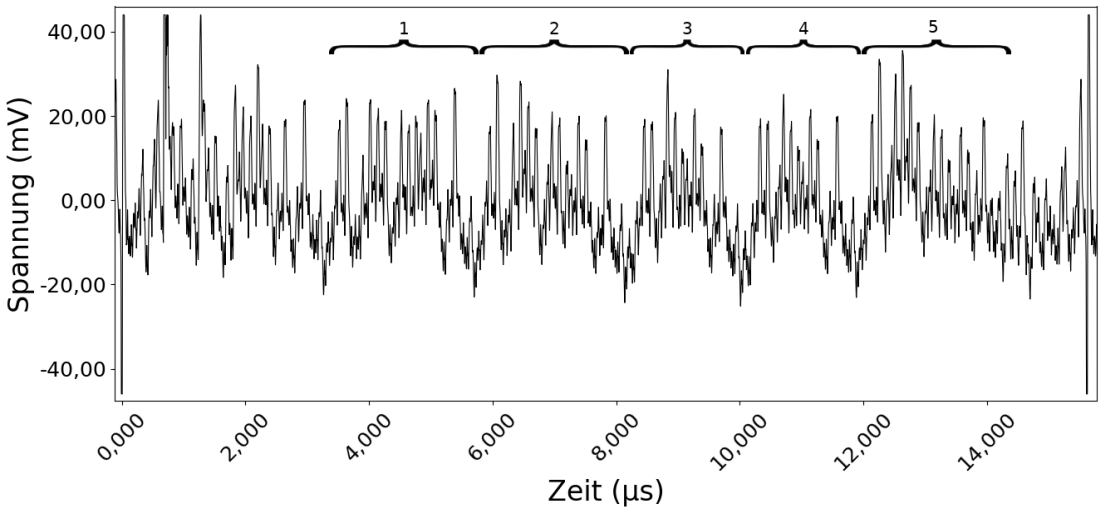


Abbildung 6.2: Aufgezeichneter Trace des Durchlaufs einer Square-and-Multiply Implementierung. Es handelt sich hierbei um das arithmetische Mittel aus 1000 Messungen.

1000 durchgeführten Messungen dar. Auf der Horizontalen ist der zeitliche Verlauf abgebildet. Die Vertikale zeigt die gemessene Spannung (siehe Abschnitt 4.2.3). Bereits in dieser Auflösung kann man auf den ersten Blick einige Merkmale erkennen.

Der Peak zum Zeitpunkt 0 entspricht dem erzeugten Triggersignal des Microcontrollers für das Oszilloskop. Das Setzen eines Ausgangs erzeugt einen sichtbaren Peak, da sich hierbei ein hoher Stromfluss verzeichnen lässt. Bei Zeitpunkt $\sim$15,6 µs ist

ebenfalls ein Peak zu verzeichnen. Hierbei handelt es sich bereits um das Triggersignal des nächsten Durchlaufs des Square-and-Multiply Verfahren, da dieses in einer Endlosschleife ausgeführt wird. Es ist somit sichergestellt, dass der abgebildete Trace einen gesamten Durchlauf enthält.

Innerhalb des Trace können Abschnitte mit einer gleichen Breite festgestellt werden. Diese sind in Abbildung 6.2 der Reihe nach durchnummeriert. Dabei sind die Abschnitte 1, 2 und 5, sowie die Abschnitte 3 und 4 von der gleichen Breite. Es kann bereits vorweggenommen werden, dass sich ein Abschnitt auf ein einzelnes Bit des Exponenten bezieht. Ein solcher Abschnitt entspricht somit einem Schleifendurchlauf innerhalb der $binary\_exp(b, n)$ Funktion der Square-and-Multiply Implementierung. In diesen Abschnitten müssen deshalb insbesondere die Operationen der Quadrierung und Multiplikation zu finden sein. Da es sich um fünf Abschnitte handelt, kann schon zu diesem Zeitpunkt die Aussage getroffen werden, dass der verwendete Exponent nicht größer als 2^5 ist. Dies resultiert aus der Annahme, dass führende Nullen in der Binärdarstellung des Exponenten bei der vorliegenden Square-and-Multiply Implementierung nicht beachtet werden.

Bei genauerer Betrachtung der einzelnen Abschnitte können sich wiederholende Muster, mit der gleichen Zeitspanne, identifiziert werden. Abbildung 6.3 zeigt hierfür einen vergrößerten Ausschnitt des Bereiches von Zeitpunkt ∼5,5 µs bis ∼10,1 µs. Dieser Ausschnitt enthält die beiden Abschnitte zwei und drei, welche sich hinsichtlich

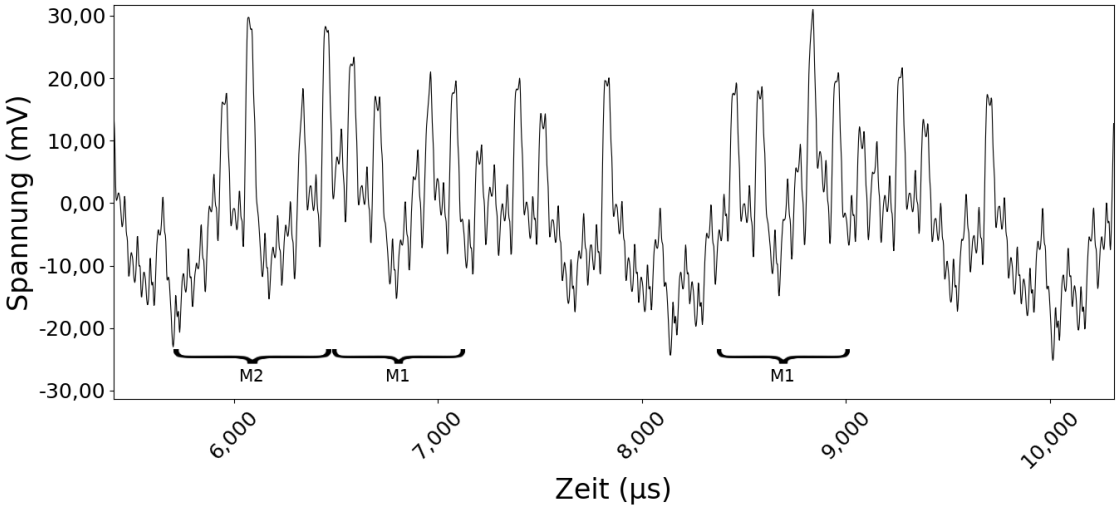


Abbildung 6.3: Vergrößerter Ausschnitt des Trace der Square-and-Multiply Implementierung. In diesem Ausschnnitt können insbesondere Muster erkannt werden, die auf gleiche Operationen bei der Verarbeitung hindeuten. Der Ausschnitt umfasst die Verarbeitung von zwei Bits des Exponenten.

ihrer Zeitspanne unterscheiden. In diesen können zwei interessante Muster festgestellt

werden, welche durch M1 und M2 entsprechend markiert sind. M1 ist in beiden Abschnitten enthalten, ebenso ist dieses Muster auch in allen anderen Abschnitten wiederzufinden. M2 hingegen ist nur in dem gezeigten Abschnitt und zusätzlich in den Abschnitten 1 und 5 enthalten. Es kann somit festgestellt werden, dass es sich bei M1 um die durchgeführte Quadrierung des Square-and-Multiply Verfahrens handelt und M2 die Multiplikation darstellt. Diese Aussage kann getroffen werden, da für jedes Bit des Exponenten eine Quadrierung durchgeführt werden muss, die Multiplikation jedoch nur bei gesetztem Bit im Exponenten durchgeführt wird.

Folglich kann festgehalten werden, dass bereits über die unterschiedliche Zeitspanne der Abschnitte Informationen preisgegeben werden. Hinter den Abschnitten der größeren Zeitspanne (Abschnitte 1, 2 und 5) verbirgt sich eine Quadrierung und eine Multiplikation. Hingegen beinhalten die Abschnitte drei und vier lediglich eine Quadrierung. Durch den Vergleich der Muster kann auf die Reihenfolge der Operationen geschlossen werden. Der Trace gibt somit Aufschluss über die durchgeführten Operationen.

Abbildung 6.4 stellt dies nochmals graphisch dar. Einzelne Operationen können

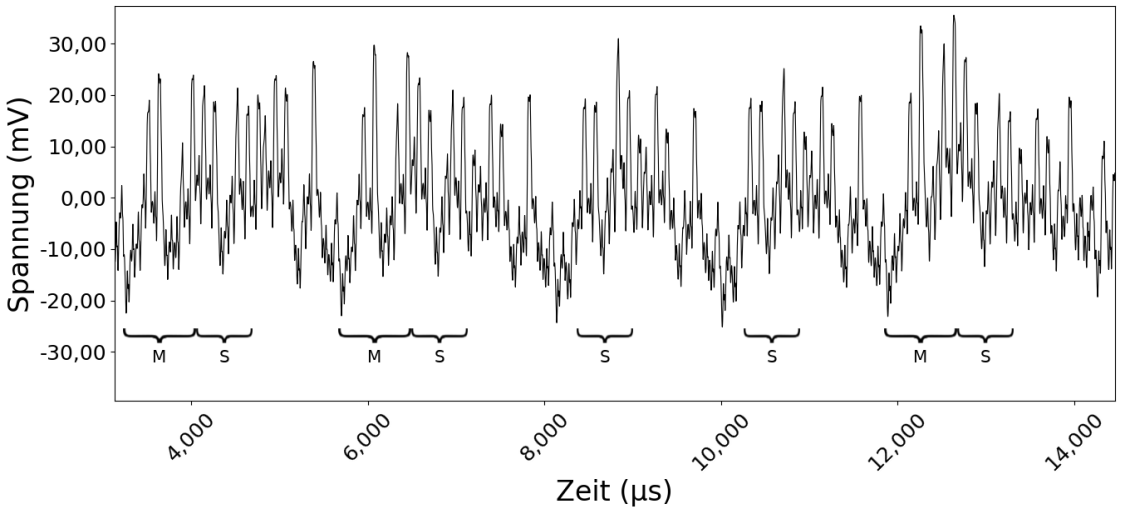


Abbildung 6.4: Durch die Erkennung von Mustern in dem Trace einer Square-and-Multiply Implementierung, können die Operationen (Quadrierung, Multiplikation) des Square-and-Multiply Verfahren ermittelt werden. Die Abbildung zeigt den relevanten Ausschnitt, in welchem die Verarbeitung der fünf Bits des Exponenten stattfindet.

hierbei direkt aus dem Trace abgelesen werden. Aus der daraus resultierenden Zeichenkette MSMSSSMS kann der Exponent abgeleitet werden. Hierfür muss man lediglich die Zeichenkette SM durch eine 1 und das Zeichen S durch eine 0 ersetzen. Bei dieser Implementierung muss jedoch noch Folgendes beachtet werden: Der Exponent wird vom LSB hin zum MSB durchlaufen. Hierdurch muss die entstehende Zeichenkette

der Quadrierungen und Multiplikationen vorher noch umgekehrt werden. Dies ist ebenfalls aus dem Trace ersichtlich, denn die resultierende Zeichenkette muss aufgrund des Algorithmus immer mit einer Quadrierung beginnen. Die Zeichenkette ist somit SMSSSMSM und entspricht nach dem Ersetzen einem Exponenten von 10011_2.

6.5.3 Ergebnis

Die vorliegende SPA konnte erfolgreich durchgeführt werden. Es wurde gezeigt, wie der Exponent des Square-and-Multiply Verfahren extrahiert werden kann. Hierfür genügt bereits die Betrachtung eines einzelnen Traces des Stromverbrauchs. Ein Angriff dieser Art kann durchgeführt werden, da die Operationen (Quadrierung, Multiplikation) des Algorithmus direkt von den einzelnen Bits des Exponenten abhängig sind und in dem Trace identifiziert werden können.

Da das Square-and-Multiply Verfahren zum Beispiel bei der RSA-Verschlüsselung eingesetzt wird, ist RSA prinzipiell anfällig für das gezeigte Verfahren. Der verwendete private RSA-Schlüssel kann auf diesem Wege extrahiert werden.

6.6 Differential Power Analysis

In diesem Abschnitt wird gezeigt, wie ein Angreifer mit Hilfe der DPA an das Schlüsselmaterial einer Verschlüsselung gelangen kann. Konkret wird der verwendete Schlüssel einer AES[2]-128 (im Folgenden einfach nur AES genannt) Implementierung extrahiert. Die vorgestellte Vorgehensweise lässt sich auf AES-192 und AES-256 mit kleinen Änderungen übertragen. Als Zielsystem wird hierbei der in Abschnitt 6.1 beschriebene Microcontroller verwendet. Dieser führt den AES-Algorithmus aus.

Da das Schlüsselmaterial sowohl bei dem Verschlüsseln als auch dem Entschlüsseln von Daten verwendet wird, sind verschiedene Szenarien für einen Angreifer denkbar:

1. Kryptosystem führt Verschlüsselung mit bekannten Klartexten aus.
2. Kryptosystem führt Verschlüsselung mit bekannten Chiffraten aus.
3. Kryptosystem führt Entschlüsselung mit bekannten Klartexten aus.
4. Kryptosystem führt Entschlüsselung mit bekannten Chiffraten aus.

In Bezug auf AES ist ein Angriff auf jeder der vier genannten Szenarien möglich. Es ist somit unerheblich, ob das kryptographische System eine Verschlüsselungsoperation oder eine Entschlüsselungsoperation ausführt. Ebenso spielt es keine Rolle, ob einem Angreifer die verwendeten Klartexte oder die Chiffrate zur Verfügung stehen. Ein Angreifer kann somit bei jedem oben aufgeführten Szenario eine DPA durchführen und an den verwendeten Schlüssel gelangen.

[2] Eine Beschreibung des AES befindet sich in Abschnitt 3.

Die weitere Beschreibung beruht auf der Annahme des ersten Szenario. Es wird davon ausgegangen, dass das kryptographische System als Operation eine AES-Verschlüsselung ausführt. Zudem wird angenommen, dass ein Angreifer beliebige Klartexte an das System übertragen kann und das Kryptosystem diese verschlüsselt. Weiterhin wird vorausgesetzt, dass der Angreifer direkten physischen Zugriff auf das kryptographische System besitzt.

6.6.1 Verwendetes Programm

Es wurden einige Voraussetzungen an das Programm gestellt, welches für diesen Testaufbau auf dem Microcontroller ausgeführt wird. Das Programm sollte ohne großen Aufwand mit unterschiedlichen Daten, hiermit sind der AES-Schlüssel und verschiedene Klartexte gemeint, verwendet werden können. Deshalb wurde darauf geachtet, dass der Schlüssel bereits zur Compile-Zeit festgelegt werden kann. Zudem kann das Programm und somit die AES-Verschlüsselung mit verschiedenen Klartexten ausgeführt werden. Hierfür bietet das entwickelte Programm die Möglichkeit der Kommunikation über die serielle Schnittstelle an (siehe Abschnitt 6.4). Eine weitere Voraussetzung bestand in der Erzeugung eines geeigneten Triggersignals für das Oszilloskop. Das Programm sollte darüber Auskunft geben, sobald die kryptographische Operation ausgeführt wird. Dies wird durch ein entsprechendes Signal an einem der Pins des Microcontrollers realisiert. In Listing 6.3 ist ein Ausschnitt des Programms zu sehen, welches auf dem Microcontroller ausgeführt wird. Es zeigt das Hauptprogramm.

```
1  #include <avr/io.h>
2  #include <string.h>
3
4  #include "../aes/aes-128/aes.h"
5  #include "../usart/usart.h"
6
7
8  #define FOSC 16000000L
9  #define BAUD 9600L
10 #define UBRR ((FOSC / (16L * BAUD)) - 1)
11
12 #define trigger_high() PORTB |= 1<<PORTB1
13 #define trigger_low() PORTB &= ~(1<<PORTB1)
14
15 int main (void) {
16     DDRB |= 1<<DDB1; // configure PORTB1 as output
17
18     uint8_t key[] = KEY;
```

```
19      uint8_t plain_text[16], cipher_text[32];
20
21      usart_init((uint8_t)UBRR);
22      while (1) {
23          usart_gets(plain_text, 16);
24          usart_puts(plain_text, 16);
25
26          trigger_high();
27          trigger_low();
28
29          memset(&cipher_text, 0, sizeof(cipher_text));
30          ecb_aes_encrypt(plain_text, 16, cipher_text, key);
31
32          usart_puts(cipher_text, 16);
33      }
34
35      return 0;
36  }
```

Listing 6.3: Verwendetes Hauptprogramm bei der Durchführung der DPA

Bei dem Hautprogramm handelt es sich um ein recht simpel aufgebautes Programm. In einer Endlosschleife werden Klartexte entgegengenommen und verschlüsselt. Das Chiffrat wird an den Absender zurückgeschickt.

Da die Kommunikation mit dem Microcontroller über die serielle Schnittstelle verläuft, muss diese dementsprechend konfiguriert werden. In den Zeilen 8-10 wird hierfür der Wert für das Baudratenregister berechnet. Dieser Wert ist sowohl von der Frequenz des Microcontrollers als auch von der gewünschten Baudrate abhängig. Durch den Aufruf von *usart_init*(·) in Zeile 21 wird die serielle Schnittstelle letztendlich mit dem berechneten Wert initialisiert.

In den Zeilen 26 und 27 wird das Triggersignal für das Oszilloskop erzeugt. Hierbei wird `PORTB1` des Microcontrollers kurzzeitig gesetzt und daraufhin wieder gelöscht. Vorher wurde der Port entsprechend als Ausgang konfiguriert (Zeile 16).

Das Empfangen des Klartextes geschieht in Zeile 23. Hierbei wird ein 16 Byte langer Klartext über die serielle Schnittstelle entgegengenommen. In Zeile 24 wird selbiger Klartext über die serielle Schnittstelle zurück übertragen. Hierdurch kann das Auftreten von möglichen Übertragungsfehlern überprüft werden. Somit wird dies lediglich für debugging-Zwecke verwendet.

In Zeile 30 wird die eigentliche AES-Verschlüsselung durchgeführt. Hierbei wird der über die serielle Schnittstelle empfangene Klartext verschlüsselt. Der verwendete Schlüssel ist dabei konstant und wird in Zeile 18 initialisiert. Das Festlegen des Schlüssels erfolgt bereits zur Compile-Zeit über ein Makro.

Die verwendete AES-Implementierung wurde nicht selbst geschrieben, stattdessen wird eine bereits bestehende Implementierung gewählt. Diese kann von [SV] bezogen werden. Hierbei handelt es sich um eine ganz gewöhnliche AES-Implementierung ohne jegliche Gegenmaßnahmen in Bezug auf Seitenkanalangriffe.

6.6.2 Durchführung

Für die Durchführung der DPA wurde sich an das in Abschnitt 4.6.2 beschriebene Verfahren gehalten. Im Folgenden wird anhand der einzelnen fünf Schritte die Vorgehensweise einer solchen Analyse erläutert. Die Schritte eins und zwei müssen lediglich einmal durchgeführt werden, die Schritte drei, vier und fünf hingegen beziehen sich nur auf einen kleinen Teil des AES Schlüssels. Diese müssen jeweils pro einzelnes Schlüsselbyte, demnach 16 mal, durchgeführt werden.

Schritt 1: Auswahl eines Zwischenergebnisses des ausgeführten Algorithmus

Ziel: *Zwischenzustand der AES-Implementierung identifizieren, welcher sich als Funktion $f(d, k)$ darstellen lässt und nur von einem Teil des Schlüssels (k) und dem Klartext (d) abhängt.*

Im ersten Schritt wird ein Zwischenzustand festgelegt, welcher sowohl von einem bekannten konstanten Datum als auch von einem Teil des Schlüssels abhängt. Aufgrund der getätigten Annahme, dass dem Angreifer ausschließlich die verschiedenen Klartexte vorliegen, handelt es sich bei dem konstanten Datum um einen Klartext. Die Funktion $f(d, k)$ muss demnach ein Zwischenergebnis der ausgeführten AES-Implementierung berechnen können, welcher aus einem Klartext und einem Teil des Schlüssels besteht. Da AES byteorientiert arbeitet und die 16 Bytes eines Blockes bis zur MixColumns Operation völlig unabhängig voneinander sind, wird als Zwischenergebnis eine Verknüpfung zwischen einem einzelnen Byte des Schlüssels und einem Byte des Klartextes gewählt. Der Klartext ist dem Angreifer hierbei bekannt. Über den Schlüssel hingegen werden zu einem späteren Zeitpunkt (Schritt 3) hypothetische Annahmen getroffen.

Als geeigneter Zwischenzustand erweist sich das Ergebnis der SubBytes Operation von der ersten Verschlüsselungsrunde (siehe Abbildung 3.1) der AES-Verschlüsselung als äußerst sinnvoll. Dieser Zustand ist lediglich von einem Byte des Schlüssels und einem Byte des Klartextes abhängig, wodurch man diesen einfach berechnen kann. Er lässt sich durch $f(d, k) = sbox(d_i \oplus k_i)$ beschreiben, wobei d_i dem i-ten Byte des Klartextes und k_i dem i-ten Byte des Schlüssels entsprechen. Eine positive Eigenschaft dieses Zustandes ist der SubBytes Operation geschuldet. Diese sorgt für eine Nichtlinearität der Daten. Mit Hilfe der festgelegten Funktion kann man

somit für ein bekanntes Klartextbyte d_i und einer hypothetischen Annahme über den verwendeten Schlüssel k_i ein Zwischenergebnis der AES-Implementierung berechnen.

In Tabelle 6.2 sind die Zwischenergebnisse in Bezug auf die in Abschnitt 6.6 beschriebenen Szenarien 1 und 2 aufgeführt. p_i entspricht dem i-ten Byte des Klartextes

Bekanntes Datum	Zwischenergebnis	Szenario
Klartext	$s_i = sbox(p_i \oplus k_i)$	1
Chiffrat	$s_i = isbox(c_i \oplus k_i)$	2

Tabelle 6.2: Überblick über die mögliche Auswahl eines Zwischenergebnisses einer AES-Implementierung in Bezug auf das Durchführen einer DPA. Ein Zwischenergebnis (s_i) kann sowohl von einem Klartextbyte (p_i), Chiffratbyte (c_i) oder Schlüsselbyte (k_i) abhängig sein.

und c_i dem i-ten Byte des Chiffrates. Bei k_i handelt es sich um das i-te Byte des hypothetischen Schlüssels, $isbox(\cdot)$ stellt die Inverse der S-Box dar. Die Auflistung ist nicht Vollständig.

Schritt 2: Messen des Stromverbrauchs

Ziel: *Durchführung der Messung des Stromverbrauchs, währenddessen der Microcontroller D verschiedene Klartexte verschlüsselt.*

Als nächstes werden tatsächliche Messungen des Stromverbrauchs durchgeführt. Hierfür wird das in Abschnitt 6.6.1 gezeigte Programm auf dem Microcontroller ausgeführt. Dadurch können verschiedene Klartexte verschlüsselt und der Stromverbrauch gemessen werden. Der Stromverbrauch wird während der Durchführung der Verschlüsselung durch das Oszilloskop aufgezeichnet.

Bei den Klartexten handelt es sich um bekannte, jedoch verschiedene Daten. Hierfür werden von seitens des Computers zufällige Klartexte generiert. Die Größe eines Klartextes beträgt 16 Byte, dies entspricht genau der Größe eines Blockes der AES-Verschlüsselung. Für diese Analyse wurden insgesamt 3000 verschiedene Klartexte erzeugt.

Die Übertragung der Klartexte zwischen Computer und Microcontroller geschieht über die vom Microcontroller bereitgestellte serielle Schnittstelle (siehe Abschnitt 6.4). Hierfür werden Nachrichten mit einer Größe von 16 Byte ausgetauscht. Somit kann letztendlich jeder einzelne Klartext durch die sich auf dem Microcontroller befindliche kryptographische Operation mit AES verschlüsselt werden. Die kryptographische Operation wird demzufolge mehrmals ausgeführt.

Zu jedem Durchlauf wird der Stromverbrauch mit Hilfe des Oszilloskops gemessen. Die Vorgehensweise einer solchen Messung ist in Abschnitt 4.2.3 bereits beschrieben. Das Oszilloskop ist derart konfiguriert, dass eine Messung aus 1024^2 Messpunkten besteht. Die durchgeführten Messungen können somit als eine Matrix $\boldsymbol{T}$ der Größe $D \times T$ mit D Traces und T Messpunkten aufgefasst werden. In diesem Fall ist $D = 3000$ und $T = 1024^2$.

In Abbildung 6.5 ist eine solche Messung des Stromverbrauchs als Beispiel aufgeführt. Auch hier kann man bereits wie bei der SPA visuelle Merkmale in dem

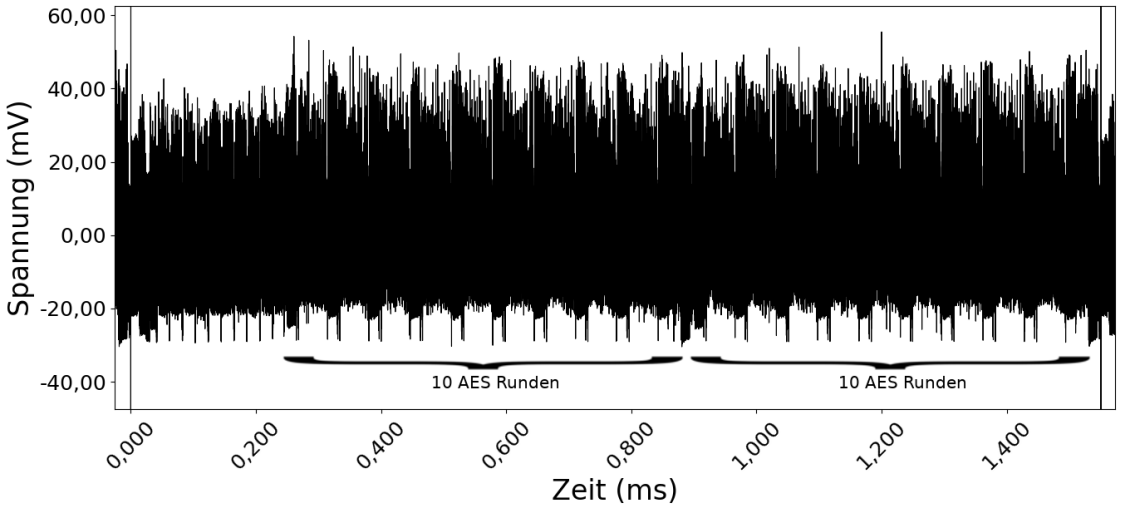


Abbildung 6.5: Aufgezeichneter Trace des Durchlaufs einer AES-128 Implementierung. Deutlich zu erkennen sind die einzelnen AES Runden.

Trace identifizieren. So sind unter anderem deutlich die einzelnen 10 AES Runden zu erkennen[3]. Ebenso können einzelne Operationen wahrgenommen werden. Abbildung 6.6 zeigt einen vergrößerten Ausschnitt einer AES Runde. Relativ klar erkennbar ist die AddRoundKey Operation der AES-Verschlüsselung. Diese führt eine XOR-Verknüpfung zwischen zwei Bytes durch. Da immer Blöcke der Größe von 16 Byte verarbeitet werden, sind infolgedessen in dem Stromverbrauch exakt 16 Peaks dafür zu verzeichnen, und zwar ein Peak pro XOR-Verknüpfung. Listing 6.4 gibt einen Einblick in die genaue Implementierung der AddRoundKey Operation.

```
227 void aes_addroundkey(uint8_t state[], const uint8_t round_key[])
228 {
229     unsigned int i;
230
```

[3] Es sind insbesondere sogar zweimal 10 AES Runden zu erkennen. Der Grund liegt in der Implementierung des Padding, welches verwendet wird, um die Klartexte auf ein Vielfaches der Blockgröße zu erweitern. Bei der verwendeten AES-Implementierung wird sich an das in Requests for Comments (RFC) 5652 beschriebene Paddingverfahren gehalten. Hierbei wird selbst für Daten, die bereits einem Vielfachen der Blockgröße entsprechen, ein Padding erzeugt.[Hou09, S. 27 f.]

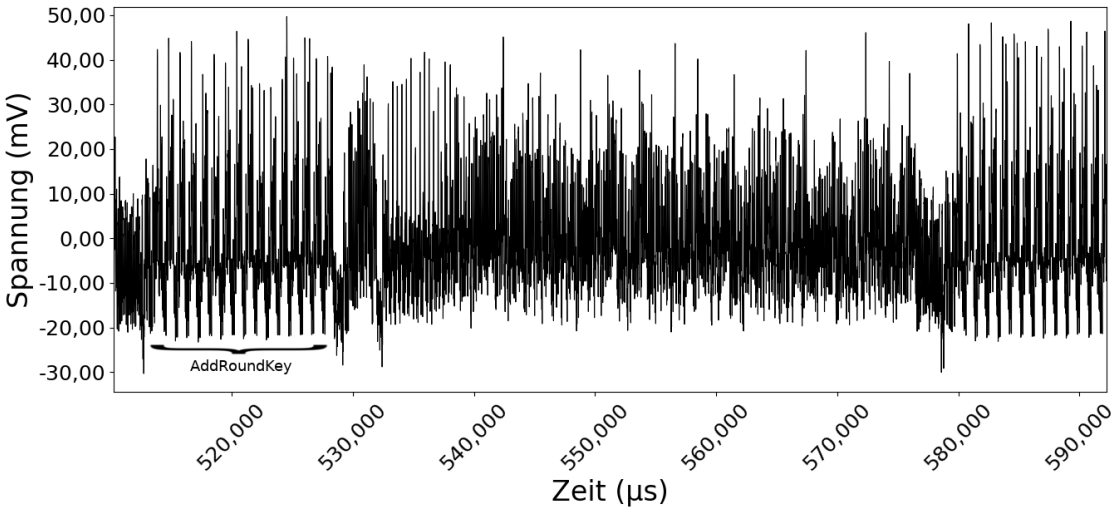


Abbildung 6.6: Vergrößerter Ausschnitt des Trace der AES-Implementierung. Der Ausschnitt umfasst eine ganze AES Runde. Sehr schön zu erkennen ist insbesondere die AddRoundKey Operation.

```
231     for (i = 0; i < 16; ++i)
232         state[i] ^= round_key[i];
233 }
```

Listing 6.4: AddRoundKey Operation der verwendeten AES-Implementierung.

Zu sehen ist die XOR-Verknüpfung zwischen den einzelnen Bytes des aktuellen Zustandes und den Bytes des Rundenschlüssels.

Diese visuellen Merkmale sind für den Erfolg der DPA nicht zwingend erforderlich. Sie zeigen jedoch, dass ein Angreifer sogar ohne jegliche Kenntnis eines kryptographischen Systems, Vermutungen über das verwendete Verschlüsselungsverfahren treffen kann. Solche Vermutungen können zum Beispiel durch das bloße Erkennen der Anzahl der Runden oder das Identifizieren einzelner bekannter Operationen getroffen werden.

Schritt 3: Berechnung hypothetischer Zwischenwerte

Ziel: *Treffen von hypothetischen Annahmen über ein Schlüsselbyte und Berechnung aller möglichen Zwischenergebnisse anhand der in Schritt 1 festgelegten Funktion $f(d, k)$.*

Nach dem Aufzeichnen der Traces, wird zu jedem bekannten Klartext $\boldsymbol{d}$, das hypothetische Zwischenergebnis für alle möglichen Schlüsselhypothesen berechnet. Hierfür wird die in Schritt 1 festgelegte Funktion $f(d, k)$ verwendet. Diese berechnet ein Zwischenergebnis für ein gegebenes Klartextbyte und einen Schlüsselbyte. Aufgrund von diesem Umstand muss der gegenwärtige Schritt mehrmals ausgeführt werden,

und zwar jeweils einmal für alle 16 Bytes des Klartextes. Dies ist nötig, da $f(d, k)$ nur von genau einem Byte des Schlüssels und Klartextes abhängt. Über den Schlüssel werden hypothetische Annahmen getroffen. Da es sich hierbei um ein Byte handelt, sind insgesamt 256 verschiedene Schlüsselhypothesen $\boldsymbol{k}$ möglich, demzufolge ist $\boldsymbol{k} = \{0, 1, 2, \ldots, 254, 255\}$. Die Ergebnismatrix $\boldsymbol{V}$ enthält somit Zeilenweise ein Zwischenergebnis des AES-Algorithmus für alle möglichen Schlüsselhypothesen zu einem bekannten Klartextbyte.

Schritt 4: Mapping der hypothetischen Zwischenwerte auf hypothetische Stromverbrauchswerte

Ziel: *Berechnete Zwischenergebnisse mit Hilfe des Hamming-Gewicht Modell auf entsprechende Stromverbrauchswerte abbilden.*

Um die in Schritt 2 aufgezeichneten Traces mit den hypothetisch berechneten Zwischenergebnissen vergleichen zu können ist ein weiterer Zwischenschritt nötig. Die in Schritt 3 berechneten hypothetischen Zwischenwerte, welche einen Zwischenzustand der ersten AES Runde widerspiegeln, sind Ergebnisse des AES-Algorithmus und stehen nicht im Zusammenhang mit den gemessenen Stromverbrauchswerten. Um diese vergleichen zu können, ist eine Abbildung zwischen den hypothetischen Zwischenergebnissen auf entsprechende hypothetische Stromverbrauchswerte nötig. Eine solche Abbildung kann zum Beispiel durch die in Abschnitt 4.4 vorgestellten Power Modelle erreicht werden. Für die durchgeführte DPA eignete sich das Hamming-Gewicht als Power Modell[4]. Die Stromverbrauchswerte können somit durch $HW(f(d, k))$ beschrieben werden.

Schritt 5: Vergleich der hypothetischen Stromverbrauchswerte mit den aufgezeichneten Traces

Ziel: *Stromverbrauchswerte unter Zuhilfenahme der Bravais-Pearson-Korrelation mit den tatsächlich aufgezeichneten Traces vergleichen.*

Im letzten Schritt werden die hypothetisch berechneten Stromverbrauchswerte aus Schritt 4 mit den tatsächlich aufgezeichneten Traces aus Schritt 2 verglichen. Dafür wird die in Abschnitt 4.5 beschriebene Pearson-Korrelation verwendet. Durch Einsetzen der entsprechenden Werte ergibt sich die in Gleichung 6.1

$$r_{ij} = \frac{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i) \cdot (t_{d,j} - \bar{t}_j)}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2 \cdot \sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2}} \tag{6.1}$$

[4] Für gewöhnlich eignet sich das Hamming-Gewicht Modell für Angriffe auf kryptographische Operationen, die in Software implementiert sind. Das Hamming-Distanz Modell hingegen eignet sich eher bei Angriffen auf Hardwareimplementierungen.

abgebildete Formel.

Durch die Pearson-Korrelation wird der Korrelationskoeffizient zwischen dem gemessenen Stromverbrauch und dem hypothetisch berechneten Stromverbrauch jeder Schlüsselhypothese berechnet. Die Berechnung wird für jeden einzelnen Zeitpunkt der Traces durchgeführt. Das Ergebnis dieser Berechnung kann in einer Matrix dargestellt werden.

Abbildung 6.7 stellt die daraus resultierenden Werte als Graphen dar. Auf der

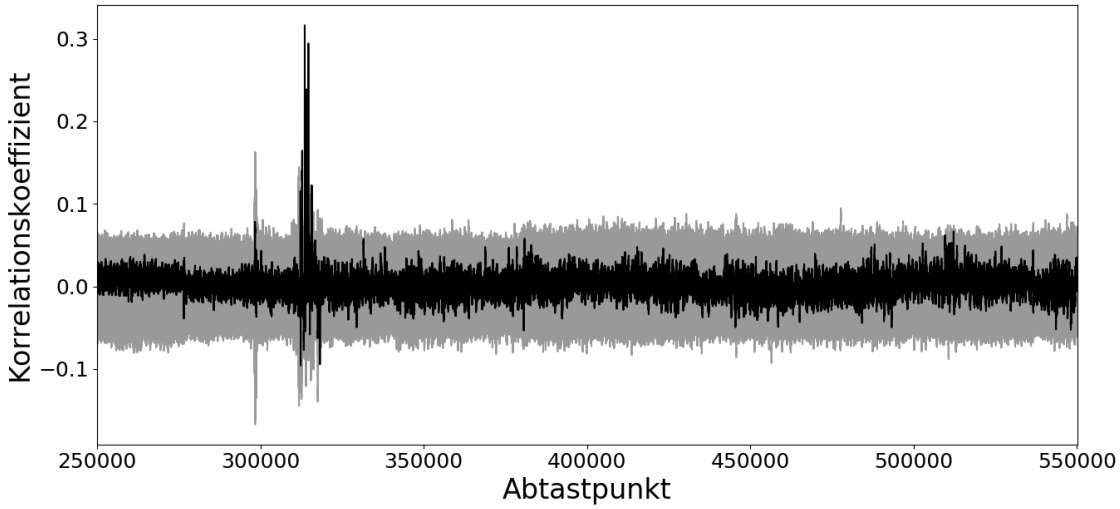


Abbildung 6.7: Es werden alle Schlüsselhypothesen, also 256 Graphen, mit den jeweiligen Korrelationskoeffizienten zu den einzelnen Abtastpunkten dargestellt. Die richtige Schlüsselhypothese ist in Schwarz abgebildet.

Horizontalen ist der zeitliche Verlauf, folglich die Messpunkte zu einem bestimmten Zeitpunkt zu erkennen. Die Vertikale gibt Auskunft über den berechneten Korrelationskoeffizienten. Abgebildet sind die einzelnen Graphen zu allen Schlüsselhypothesen. Demnach stellt die Abbildung 256 Graphen dar. Der Graph der richtigen Schlüsselhypothese ist dabei in Schwarz angedeutet, die Graphen aller anderen Schlüsselhypothesen sind Grau abgebildet. Der schwarze Graph sticht als einziger Graph deutlich hervor. Dies stellt ein Indiz für den Erfolg der DPA. Zudem gibt die Abbildung ebenfalls darüber Auskunft, zu welchem Zeitpunkt die erste Runde der AES-Verschlüsselung ausgeführt wird. Da auf ein Zwischenergebnis der ersten Runde korreliert wird (siehe Schritt 1), ist davon auszugehen, dass diese ca. bei Messpunkt 310000 beginnt.

Die Auswertung kann ebenfalls über die Betrachtung der maximalen Korrelationskoeffizienten jeder einzelnen Schlüsselhypothese erfolgen. Diese sind in Abbildung 6.8 dargestellt. In dieser Abbildung teilt sich die Horizontale auf 256 Teile auf, welche für die einzelnen Schlüsselhypothesen stehen. Auf der Vertikalen wird der Korrelationskoeffizient abgebildet. Hierbei ist zu beachten, dass es sich um den maximalen

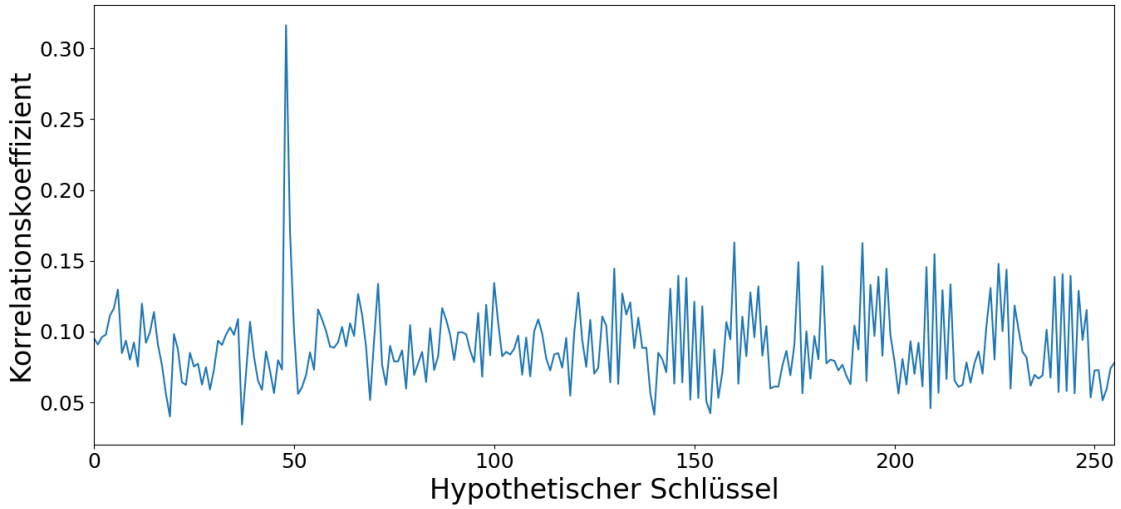


Abbildung 6.8: Es wird der jeweilige maximale Korrelationskoeffizient zu jeder Schlüsselhypothese abgebildet. Der Graph hat somit keinen Bezug zum zeitlichen, demnach zu den Abtastpunkten.

Korrelationskoeffizienten zu der Horizontal angegebenen Schlüsselhypothese zu einem beliebigen Messzeitpunkt handelt. Das heißt, die Abbildung gibt keinerlei Informationen über das zeitliche Verhalten. Es kann lediglich abgelesen werden, dass eine bestimmte Schlüsselhypothese zu irgendeinem Messzeitpunkt einen maximalen Korrelationskoeffizienten k aufweist. Aus dieser Abbildung kann z.B. geschlossen werden, dass die Schlüsselhypothese 50, zu irgendeinem Messzeitpunkt, einen Korrelationskoeffizienten k von ca. 0.31 aufweist und es sich hierbei um den maximalen Wert über den gesamten Messzeitraum handelt.

6.6.3 Dauer

Die Dauer für die Durchführung der DPA hängt maßgeblich von den folgenden zwei Faktoren ab: Erstens von der benötigten Zeit für das Aufzeichnen und die Übertragung einzelner Traces von dem Oszilloskop zu dem Computer. Zweitens von der benötigten Zeit für die Berechnung des Korrelationskoeffizienten und somit von der Rechenleistung des Computers.

Für die Aufzeichnung und die Übertragung eines einzelnen Trace, bestehend aus 1024^2 Messpunkten, werden ca. 6 Sekunden benötigt. Demnach werden für Schritt 2 der unter Abschnitt 6.6.2 beschriebenen Vorgehensweise ca. 5 Stunden für 3000 Traces benötigt. Der Flaschenhals ist hierbei tatsächlich die Übertragung zwischen Oszilloskop und Computer, da es sich bei der am Oszilloskop befindlichen USB-Schnittstelle lediglich um eine USB 1.0 Schnittstelle handelt und dadurch eine maximale Datenübertragungsrate von 1 MB/s möglich ist.

Das Berechnen des Korrelationskoeffizienten stellt einen weiteren zeitintensiven Schritt dar. Die Berechnung an sich ist keine komplexe Operation, jedoch ist die Anzahl der Ausführung ausschlaggebend. Auf dem Intel Xeon Prozessor E5-2630L benötigt die Berechnung ca. 8 Stunden. Im Allgemeinen stellt dies natürlich kein Problem dar. Bei Testumgebungen hingegen möchte man gerne schnell Ergebnisse sehen. Aufgrund dessen wurde die Berechnung des Korrelationskoeffizienten auf eine GPU ausgelagert, genaueres wurde bereits in Abschnitt 5.2.2 erläutert. Durch die Umstellung der Berechnung auf einer GPU kann einiges an Zeit eingespart werden. Die Berechnung dauert dadurch vielmehr nur noch ca. 5 Minuten anstatt 8 Stunden.

6.6.4 Ergebnis

Die vorliegende DPA konnte erfolgreich durchgeführt werden. Es wurde gezeigt, wie der Schlüssel einer AES-Verschlüsselung extrahiert werden kann. Der Angriff bezog sich auf eine konkrete Implementierung des AES-Algorithmus, die auf einem Microcontroller ausgeführt wurde. Für die Durchführung wurden 3000 unterschiedliche Klartexte durch den Microcontroller verschlüsselt und der Stromverbrauch durch ein Oszilloskop aufgezeichnet. Durch hypothetische Annahmen über den Schlüssel und den tatsächlich aufgezeichneten Traces konnten Korrelationen berechnet und der verwendete AES Schlüssel ermittelt werden. Durch eine solche Korrelationsberechnung können Störfaktoren wie zum Beispiel Rauschen herausgerechnet werden.

7 Fazit

In dieser Arbeit wurde eine Laborumgebung für Power Analysis Attacks aufgebaut. Anhand eines Testaufbaus, bestehend aus einem Microcontroller, welcher als kryptographisches System fungiert, wurde demonstriert, wie eine Analyse per SPA und DPA durchgeführt werden kann. Besonders die DPA stellt dabei ein effektives Verfahren bereit. Es wurde gezeigt, wie damit der geheime Schlüssel einer konkreten Implementierung des Verschlüsselungsverfahrens AES ermittelt werden kann. Hierbei hat es sich um AES-128 gehandelt. Durch diesen Testaufbau wird die prinzipielle Verwundbarkeit solcher Systeme und die Funktionstüchtigkeit der Testumgebung gezeigt.

Es wurde gezeigt, dass für solche Angriffe keine besonders hochwertige Ausstattung nötig ist. Die verwendete Hardware ist für jedermann erhältlich und kann für einige hundert Euro erworben werden. Die Durchführung einer DPA beschränkt sich somit nicht auf bestimmte Personen, sondern kann von einer beliebigen in diesem Bereich versierten Person durchgeführt werden.

Ebenfalls wurde Auskunft über die Dauer der durchgeführten DPA gegeben. Durch den Einsatz einer GPU konnte die DPA innerhalb von knapp über 5 Stunden erfolgreich durchgeführt werden. Es kann jedoch keine allgemeingültige Aussage getroffen werden, da die Dauer einer DPA von vielerlei Faktoren abhängig ist. Insbesondere das Oszilloskop und die Rechenleistung des Computers sind dabei wichtige Faktoren. Die Performance kann durch ein Oszilloskop mit einer besseren Schnittstelle erhöht werden. Jedoch hängt die Dauer ebenso von der Geschwindigkeit des anzugreifenden kryptographischen Systems ab. Sind hierbei lange Reaktionszeiten des Kryptosystems zu erwarten, so verlängert sich die Dauer der DPA erheblich.

Abschließend kann der Entschluss gefasst werden, dass Seitenkanalangriffe und im speziellen Power Analysis Attacks eine reale Gefahr auf Microcontroller darstellen. Sie erlauben kryptographische Systeme anzugreifen und müssen deshalb beachtet werden.

A Allgemein

A.1 Beispielprogramm, für das Prinzip von Power Analysis Attacks

```
1  #define __SFR_OFFSET 0
2  #include <avr/io.h>
3
4  .global main
5
6  main:
7      cli;
8
9      ldi r16, 0x02
10     out DDRB, r16    ; configure PB1 as output
11
12 loop:
13
14     ldi r16, 0x02
15     out PORTB, r16   ; set PB1 - trigger high
16
17     nop
18     nop
19     nop
20     nop
21     nop
22     ldi r16, VAL
23     nop
24     nop
25     nop
26     nop
27     nop
28     sts 0x0100, r16
29     nop
30     nop
31     nop
32     nop
33     nop
```

```
34
35      ldi r16, 0x00
36      out PORTB, r16   ; clear PB1 - trigger low
37
38      rjmp loop
```

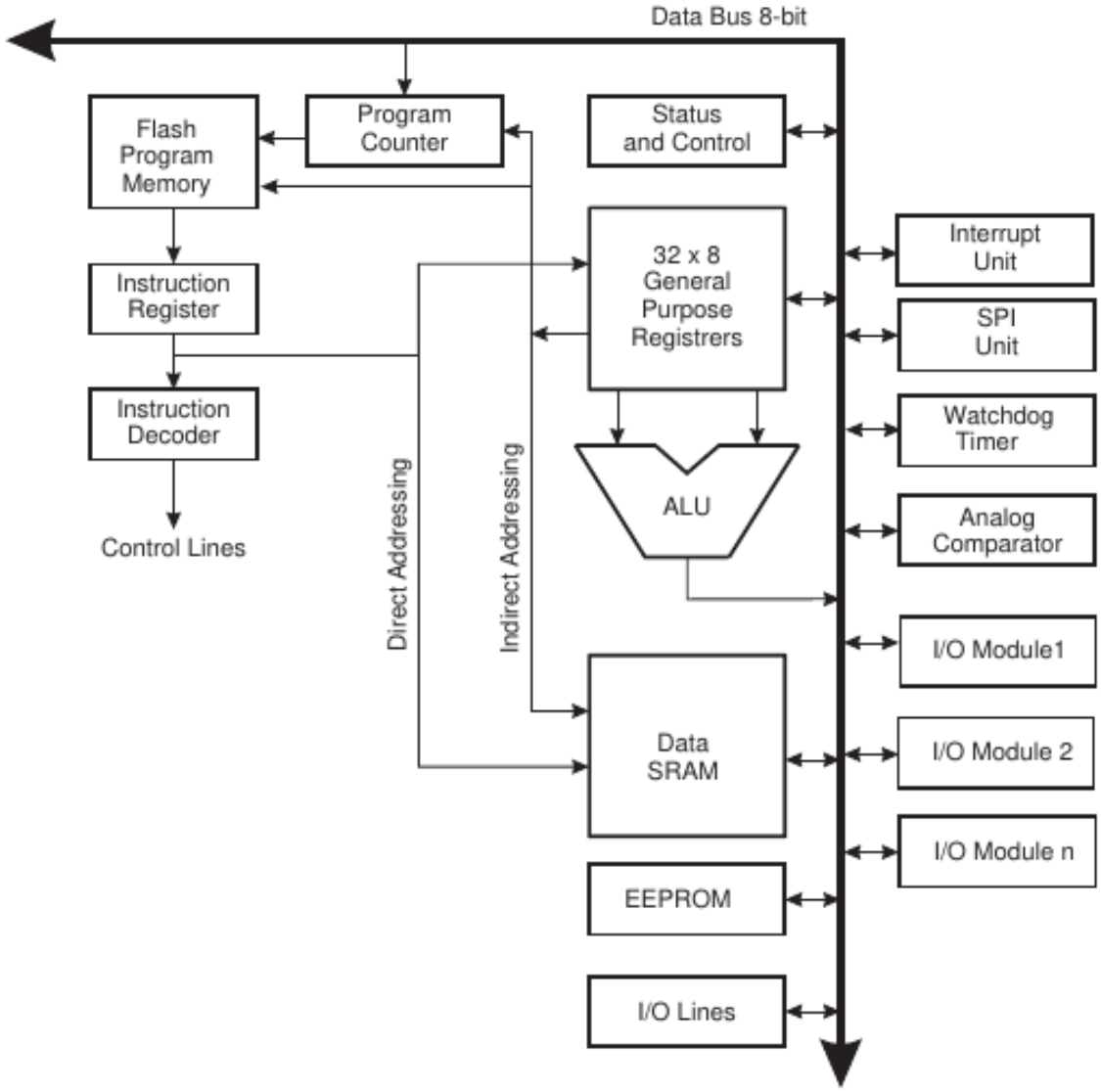
Listing A.1: Beispielprogramm, um die Abhängigkeit des Stromverbrauchs zu der Anzahl der gesetzten Bits auf dem Datenbus eines Microcontrollers zu verdeutlichen.

A.2 AES - S-Box

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Tabelle A.1: Es wird die S-Box des AES-Algorithmus dargestellt. Die Werte sind in Hexadezimaler Darstellung angegeben. Beispiel: $sbox(0x17) = f0$.

A.3 Atmel AVR Architektur



Bildquelle: [Atm15, S. 9]

Abbildung A.1: Darstellung der Atmel AVR Architektur.

Literaturverzeichnis

[And+05] Ross Anderson et al. *Cryptographic processors – a survey*. UCAM-CL-TR-641. University of Cambridge, Computer Laboratory, Aug. 2005. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-641.pdf.

[Atm15] Atmel. *ATmega48A/PA/88A/PA/168A/PA/328/P datasheet*. 2015. URL: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf.

[BB03] David Brumley and Dan Boneh. "Remote timing attacks are practical". In: *Proceedings of the 12th USENIX Security Symposium*. 2003. URL: http://www.usenix.org/events/sec03/tech/brumley/brumley.pdf.

[BG11] Alex Biryukov and Johann Großschädl. *Cryptanalysis of the Full AES Using GPU-Like Special-Purpose Hardware*. Cryptology ePrint Archive, Report 2011/710. http://eprint.iacr.org/2011/710. 2011.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. *Biclique Cryptanalysis of the Full AES*. Cryptology ePrint Archive, Report 2011/449. http://eprint.iacr.org/2011/449. 2011.

[Coma] Wikimedia Commons. *CMOS Inverter*. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/0/03/Inverter1.svg/2000px-Inverter1.svg.png.

[Comb] Wikimedia Commons. *Correlation examples*. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d4/Correlation_examples2.svg/2000px-Correlation_examples2.svg.png.

[Eis+08] Thomas Eisenbarth et al. "On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme". In: *Advances in Cryptology — CRYPTO 2008*. Springer, 2008, pp. 203–220. ISBN: 978-3-540-85174-5. DOI: 10.1007/978-3-540-85174-5_12. URL: http://dx.doi.org/10.1007/978-3-540-85174-5_12.

[Gen+16a] Daniel Genkin et al. *ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs*. Cryptology ePrint Archive, Report 2016/129. http://eprint.iacr.org/2016/129. 2016.

[Gen+16b] Daniel Genkin et al. *ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels*. Cryptology ePrint Archive, Report 2016/230. http://eprint.iacr.org/2016/230. 2016.

[Hou09] R Housley. *Cryptographic Message Syntax (CMS)*. RFC 5652. 2009. URL: https://www.ietf.org/rfc/rfc5652.txt.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology — CRYPTO '99*. Springer, 1999, pp. 388–397. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_25. URL: http://dx.doi.org/10.1007/3-540-48405-1_25.

[Knu81] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms (2^{nd} Edition)*. Addison-Wesley Pub (Sd), 1981. ISBN: 0201038226.

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology — CRYPTO '96*. Springer, 1996, pp. 104–113. ISBN: 978-3-540-68697-2. DOI: 10.1007/3-540-68697-5_9. URL: http://dx.doi.org/10.1007/3-540-68697-5_9.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN: 0387308571.

[MOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN: 9780849385230.

[Nat03] Committee on National Security Systems. *CNSS Policy No. 15, Fact Sheet No. 1: National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information*. 2003. URL: http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf.

[RIG13] RIGOL. *Programming Guide*. RIGOL Technologies, Inc. June 2013. URL: http://beyondmeasure.rigoltech.com/acton/attachment/1579/f-0012/0/-/-/-/-/file.pdf.

[SV] Filippo Sironi and Matteo Villa. *Implementation of the Advanced Encryption Standard with a 128 bits key and Electronic Code Book mode of operation*. URL: https://github.com/filipposironi/aes128-byte-oriented/tree/master/x86.

[WWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. "Improving Differential Power Analysis by Elastic Alignment". In: *Topics in Cryptology — CT-RSA 2011: The Cryptographers' Track at the RSA Conference*. Springer, Feb. 4, 2011, pp. 104–119. ISBN: 978-3-642-19074-2. DOI: 10.1007/978-3-642-19074-2_8. URL: https://doi.org/10.1007/978-3-642-19074-2_8.

[ZF05] YongBin Zhou and DengGuo Feng. *Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing*. Cryptology ePrint Archive, Report 2005/388. http://eprint.iacr.org/2005/388. 2005.

Alle Weblinks wurden am 28. Juli 2017 auf ihre Erreichbarkeit geprüft.

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und ist noch nicht veröffentlicht worden. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben wird.

Steinfurt, 31. Juli 2017

Erich Klundt