DPA-Analyse von Implementierungen symmetrischer kryptographischer Algorithmen

Diplomarbeit
ausgeführt am

Lehrstuhl für Datenverarbeitung
Technische Universität München
Prof. Dr.-Ing. Klaus Diepold

von

Hermann Seuschek

Betreuer:

Dipl.-Ing. Michael Pramateftakis
Dr. Torsten Schütze (Siemens AG, CT IC 3)

Beginn: 20.10.2004
Abgabe: 20.04.2005

Zusammenfassung

Seitenkanalattacken stellen eine ernsthafte Bedrohung von kryptographischen Systemen, die auf Chipkarten oder Embedded Systems basieren, dar. Die dabei analysierte Seitenkanalinformation wird von den physikalischen Systemen, auf denen kryptographische Algorithmen ausgeführt werden, an die Umgebung abgegeben. Mit Hilfe von statistischen Methoden ist es möglich, aus der Seitenkanalinformation geheimes Schlüsselmaterial zu gewinnen. Eine der zurzeit mächtigsten Seitenkanalattacken ist die *Differential Power Analysis (DPA)*, bei der die Stromaufnahme einer Chipkarte oder eines Embedded Systems während der Ausführung einer kryptographischen Operation analysiert wird.

In dieser Diplomarbeit wird ein bestehender DPA-Messaufbau erweitert und es werden DPA-Analysen von Implementierungen symmetrischer kryptographischer Algorithmen auf verschiedenen Hardwareplattformen durchgeführt. Untersucht werden Implementierungen des DES-Algorithmus und des AES-Algorithmus. Als Hardwareplattformen kommen die zwei Mikrocontroller ATmega32 und ST10F168 sowie die Chipkarten BasicCard ZC 3.3, BasicCard ZC 4.5D und Starcos SPK 2.2 zum Einsatz.

Schwerpunkte der Diplomarbeit sind messtechnische Maßnahmen zur Verbesserung der Signalqualität, Erweiterung und Optimierung der Auswerteskripte, Entwicklung einer Platine für den ATmega32 Mikrocontroller und Schaffung der Möglichkeit, Messungen an Chipkarten vorzunehmen.

Danksagung

An dieser Stelle möchte ich allen, die mich bei der Entstehung der Diplomarbeit unterstützt haben, ganz herzlich danken.

Ich danke Dr. Torsten Schütze von der Siemens AG für seine außerordentliche Hilfsbereitschaft und Geduld bei Problemen und Fragen jeder Art, für Verbesserungsvorschläge bezüglich der Ausarbeitung und besonders für die wertvollen Tipps im Bereich LaTeX. Herrn Dr. Bernd Meyer möchte ich für die hilfreichen Anregungen bei der praktischen Durchführung und für seine Hilfe beim Schreiben der Arbeit danken. Herrn Michael Pramateftakis vom Lehrstuhl für Datenverarbeitung der TU München danke ich für die Vermittlung der Diplomandenstelle, für die Bereitstellung des Notebooks und für die gute Zusammenarbeit.

Für das gute Arbeitsklima und die große Hilfsbereitschaft möchte ich mich beim Leiter des Kompetenzfeldes Kryptographie der Siemens AG, Abteilung CT IC 3, Dr. Erwin Heß sowie bei den Mitarbeitern Anton Kargl und Dr. Markus Dichtl bedanken.

Besonders danke ich auch meinen Eltern für ihre Unterstützung während meiner gesamten Ausbildungszeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Kapitel 1.

Einleitung

In einer Zeit, in der immer mehr Geschäftsprozesse auf elektronischem Wege abgewickelt werden und das Internet in vielen Lebensbereichen nicht mehr wegzudenken ist, ist es sehr wichtig, sich mit der Sicherheit von elektronischen Kommunikationssystemen auseinander zu setzen. Das Verlangen der Menschheit, auf sicherem Wege Informationen auszutauschen, ist sehr alt. Schon Gaius Julius Cäsar verwendete eine einfache Chiffre, bei der die Buchstaben des Alphabets zyklisch verschoben werden. Diese Form der Verschlüsselung bietet jedoch nach heutigem Erkenntnisstand keine ausreichende Sicherheit. Die Lehre der Verschlüsselung hat sich seit der Römerzeit bedeutend weiterentwickelt. Es wurden verschiedene mathematische Methoden und Algorithmen geschaffen, die innerhalb gewisser Modelle beweisbar sicher sind und deren Sicherheit ausschließlich auf der Geheimhaltung eines Schlüssels basiert (Kerkhoffs Prinzip). Heutzutage sind die zugrundeliegenden Algorithmen zwar hinreichend sicher, jedoch können Gesamtsysteme durch ein ungeschicktes Protokoll oder eine ungeschickte Implementierung eines Algorithmus das Ausspionieren eines geheimen Schlüssels ermöglichen.

In den letzten Jahren wurden Techniken entwickelt, die es ermöglichen, einen geheimen Schlüssel zu finden, indem man sich so genannte Seitenkanalinformationen zunutze macht. Seitenkanäle, wie z. B. die Stromaufnahme oder die Berechnungszeit eines Prozessors, tragen Informationen über den geheimen Schlüssel, die mit verschiedenen Methoden extrahiert werden können. Eine der mächtigsten bekannten Methoden zur Auswertung von Seitenkanalinformation ist die *Differential Power Analysis* (DPA). Dabei wird die Stromaufnahme eines Mikroprozessors während der Ausführung eines kryptographischen Algorithmus mehrmals gemessen, und die dabei gewonnenen Daten werden mit statistischen Methoden ausgewertet.

Bei der Siemens AG, Corporate Technology, Information and Communication, Security Technologies (CT IC 3) befindet sich ein Messstand zur Durchführung von Seitenkanalangriffen. Im Rahmen dieser Diplomarbeit soll dieser Messstand weiterentwickelt und verbessert werden. Der Messstand dient dem Evaluieren von Implementierungen kryptographischer Algorithmen, die bei der Siemens AG und Geschäftspartnern der Siemens AG entstanden sind. So kann man begleitend zur Implementierung von Algorithmen mögliche Schwachstellen frühzeitig entdecken und den Entwicklungsprozess beschleunigen. Die rechtzeitige Evaluierung im Haus kann auch die vorgeschriebene Evaluierung durch externe Stellen und die anschließende Zertifizierung durch das Bundesamt für Sicherheit in der Informationstechnik (BSI) beschleunigen, da man mögliche Beanstandungen schon im Voraus verhindern kann.

Die Diplomarbeit umfasst das Erweitern der Software zur Auswertung der Daten und das Verbessern des Messsetups für das Aufzeichnen der Seitenkanalinformation. Vor Beginn der Diplomarbeit war bereits ein Angriff auf einzelne S-Boxen einer DES-Implementierung auf dem ST10F168-Mikrocontroller möglich. Dieser grundlegende DPA-Angriff wird im Laufe der Diplomarbeit erweitert und optimiert. In der Diplomarbeit sollen auch Angriffe auf wei-

tere Hardwareplattformen durchgeführt werden. Als zweite Hardwareplattform wurde der ATmega32-Mikrocontroller gewählt, der mit einer einfachen externen Beschaltung auskommt. Dadurch ist es möglich eine Platine aufzubauen, auf der sich nur wenige Komponenten befinden, die die Messung stören könnten. Zusätzlich zu den zwei Mikrocontrollern sollen verschiedene Chipkarten untersucht werden. Für die Evaluierung von Chipkarten wird eine spezielle Schaltung benötigt, die das Oszilloskop im richtigen Moment triggert. Diese Schaltung wird ebenfalls im Rahmen dieser Diplomarbeit entwickelt. Als untersuchte Chipkarten kommen zwei low-cost Chipkarten der Firma Zeitcontrol und eine ältere Chipkarte von Giesecke und Devrient zum Einsatz. Die bestehenden Matlab-Skripte werden optimiert und an die zusätzlichen Hardwareplattformen angepasst. Neben dem Angriff auf Implementierungen des DES-Algorithmus werden die Skripte so erweitert, dass auch ein Angriff auf den AES-Algorithmus möglich wird.

Kapitel 2 gibt einen Überblick über die bekannten Seitenkanäle und die möglichen Angriffsszenarien. Der Schwerpunkt liegt dabei auf der DPA, da diese Form eines Seitenkanalangriffs Gegenstand dieser Diplomarbeit ist.

In Kapitel 3 wird der Messstand allgemein vorgestellt und es wird gezeigt, was man beim Aufbau von Messschaltungen beachten muss, damit man qualitativ hochwertige Messdaten aufzeichnen kann. Des Weiteren werden in diesem Kapitel die verwendeten und weiterentwickelten Matlab-Skripte grob vorgestellt.

Kapitel 4 beschreibt den Messaufbau für den ATmega32-Mikrocontroller. Dabei wird ein Überblick über die Prozessorarchitektur und die verwendeten Entwicklungstools gegeben. Die verwendete Messschaltung wird ebenfalls vorgestellt. Analysen von ausgewählten Messdaten runden das Kapitel ab.

Kapitel 5 widmet sich dem ST10F168-Mikrocontroller. Auch hier wird ein Überblick über die Hardwarearchitektur des Mikrocontrollers gegeben und kurz auf die Entwicklungstools eingegangen. Am Ende des Kapitels werden die Ergebnisse von verschiedenen DPA-Analysen dargestellt.

Kapitel 6 zeigt, welche Strategie beim Messen der Stromaufnahme von Chipkarten verfolgt wird. In der Einleitung des Kapitels wird, so weit es für das Verständnis nötig ist, der Kommunikationsablauf zwischen Chipkarte und Chipkartenterminal beschrieben. Anschließend wird der entwickelte Lösungsansatz zur Erzeugung eines Triggersignals besprochen. Am Ende des Kapitels werden die Ergebnisse von verschiedenen DPA-Analysen der untersuchten Chipkarten präsentiert.

Kapitel 2.

Konzept der Seitenkanalattacken

2.1. Die Problematik der Seitenkanäle

Kryptographische Algorithmen, welche hinreichend auf ihre Sicherheit untersucht wurden und deshalb als mathematisch sicher gelten, können bei der praktischen Implementierung ihre Sicherheit einbüßen. Bei der abstrakten mathematischen Untersuchung eines Algorithmus wird von einem Modell ausgegangen, bei dem physikalische Eigenschaften nicht berücksichtigt werden. Der Algorithmus wird als mathematisches Objekt mit definiertem Ein-Ausgabe-Verhalten betrachtet. Im Inneren dieses abstrakten Modells wird der Klartext P auf einen Chiffretext C in Abhängigkeit von einem geheimen Schlüssel K abgebildet. Die Abbildungsvorschrift kann und soll sogar allgemein bekannt sein, einzig und allein der geheime Schlüssel garantiert die Sicherheit eines kryptographischen Verfahrens (Kerkhoffs Prinzip).

Unbeachtet bleiben bei dieser abstrakten Betrachtungsweise so genannte *Seitenkanäle*, die bei einer praktischen Implementierung zwangsläufig auftreten. In der Praxis wird der Algorithmus auf einem Mikroprozessor bzw. in einer fest verdrahteten Logikschaltung abgearbeitet. Die kryptographischen Operationen werden also in Form von physikalischen Vorgängen ausgeführt. In digitalen elektronischen Rechnern kommt es dabei zu Schaltvorgängen in Logikgattern, die von außen z. B. als Änderungen in der Stromaufnahme wahrgenommen werden können. Der Stromverbrauch ist also ein möglicher Seitenkanal, über den geheime Informationen wie z. B. der verwendete Schlüssel preisgegeben werden können.

In den zwei Arbeiten von Paul Kocher et al. [Koc96] und [KJJ99] wurden erstmalig einfach umzusetzende Methoden vorgestellt, die sich Seitenkanalinformation zunutze machen, um an einen geheimen Schlüssel zu gelangen. Seither wurde auf dem Gebiet der Seitenkanalattacken viel Forschung betrieben, einen Überblick über die Forschungsergebnisse bieten die Aufsätze [HJMS00, QK02] sowie die Proceedings der CHES-Konferenzen [KP99, KP00, KNP01, KKP02, WKP03, JQ04].

2.2. Verbreitete Seitenkanäle

Wie in Abbildung 2.1 zu sehen ist, unterscheidet man im Wesentlichen vier mögliche Seitenkanäle:

- Die *Ausführungszeit* von kryptographischen Algorithmen, soweit sie von den verarbeiteten Daten abhängt.
- Die *Stromaufnahme* $i(t)$, welche im Zusammenhang mit den ausgeführten Rechenoperationen, aber auch mit den berechneten Datenworten steht.

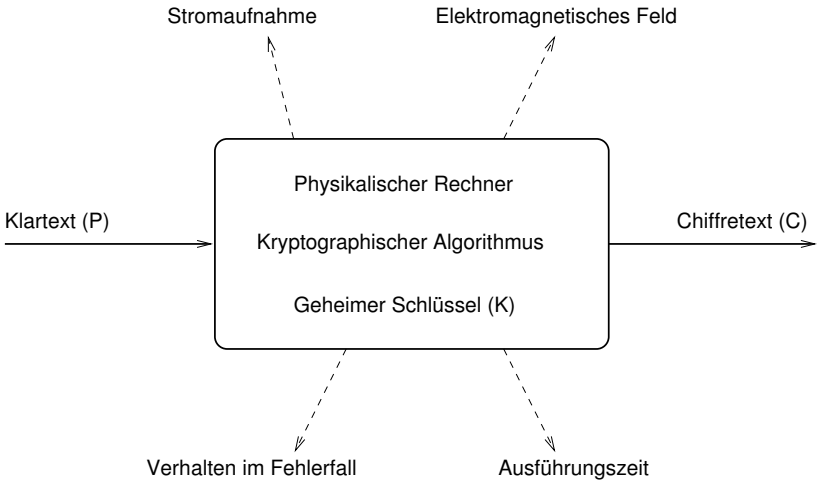

Abbildung 2.1: Verbreitete Seitenkanäle

- Die *Emission elektromagnetischer Strahlung*, die sehr eng mit der Stromaufnahme gekoppelt ist, da beide Seitenkanäle die gleiche Ursache, nämlich Schaltvorgänge der digitalen Schaltungen, haben.

- Das *Verhalten im Fehlerfall*, bei dem ein zufälliger bzw. absichtlich herbeigeführter Rechenfehler geheime Information preisgeben kann.

2.2.1. Ausführungszeit

Die Ausführungszeit kann je nach Implementierung eines Algorithmus eine Datenabhängigkeit besitzen. Ein kleines anschauliches Beispiel soll zeigen, wie eine ungeschickte Implementierung eines Algorithmus dessen Sicherheit aushebeln kann. Man stelle sich eine Passwortabfrage vor, bei der die Eingabe eines zehnstelligen numerischen Passwortes erforderlich ist. Ohne das Ausnutzen möglicher Seitenkanäle müsste man für eine Brute-Force-Attacke im Mittel $5 \cdot 10^9$ Passwörter durchprobieren, um das richtige Passwort zu erhalten. Wenn alle möglichen Passwörter mit der gleichen Wahrscheinlichkeit auftreten und man die Passwörter von Hand durchprobiert, würde das eine sehr lange Zeit in Anspruch nehmen. Die beispielhafte Implementierung (Listing 2.1) der Passwortabfrage hat jedoch eine entscheidende Schwachstelle: Der Vergleich zwischen eingegebenem Passwort und dem gespeicherten Passwort `streng_geheim` erfolgt in einer Schleife, die, sobald ein falsches Zeichen im eingegebenen Passwort entdeckt wird, mit einer Fehlermeldung verlassen wird. Dabei kommt ein Zusammenhang zwischen der Ausführungszeit und dem geheimen Passwort zustande. Die Zeit, die verstreicht, bis nach der Eingabe eines falschen Passworts eine Fehlermeldung ausgegeben wird, zeigt, wie viele Ziffern richtig eingegeben wurden, bis die erste falsche Ziffer auftritt. Mit Hilfe eines solchen Zusammenhangs hat ein potentieller Angreifer ein leichtes Spiel. Er muss lediglich von vorne beginnend für jede Ziffer alle zehn möglichen Werte ausprobieren. Der Wert, bei der die Ausgabe der Fehlermeldung am längsten verzögert wird, ist der richtige, und man kann mit der nächsten Ziffer fortfahren. Mit dieser Methode reduziert man die Anzahl der benötigten Versuche auf maximal 100.

Listing 2.1: Ungünstige Passwortabfrage

```
const char streng_geheim[] = "0815471125";

printf("Passwort␣eingeben:␣");
scanf("10s", input);
for (i = 0; i < strlen(streng_geheim); i++) {
if (streng_geheim[i] != input[i]) {
    printf("Falsches␣Passwort.\n");
    return FALSCH;
    }
}
printf("Passwort␣korrekt.\n");
return RICHTIG;
```

Dies ist nur ein triviales Beispiel, um die Problematik von ungünstig implementierten Algorithmen aufzuzeigen. Ähnliche Probleme treten jedoch immer wieder in einer weniger offensichtlichen Form bei der Implementierung von Protokollen und Algorithmen auf.

2.2.2. Stromaufnahme

Die Stromaufnahme eines Prozessors besteht aus einem statischen und einem dynamischen Teil. Die statische Stromaufnahme, verursacht durch Leckströme, ist in der Regel vernachlässigbar klein. Die dynamische Stromaufnahme steht in direktem Zusammenhang mit den verarbeiteten Daten und den ausgeführten Befehlen. Prozessoren arbeiten einen Algorithmus Schritt für Schritt ab. Dabei werden die Rechenschritte in der Regel von einem globalen Takt gesteuert. Bei jeder Taktflanke wechselt die Schaltung ihren inneren Zustand, was Schaltströme und Ladeströme von ungewollten (parasitären) Kapazitäten zur Folge hat.

Die größten parasitären Kapazitäten C_p kann man bei den relativ langen Busleitungen finden, die verschiedene Schaltungsteile miteinander verbinden und über die Zwischenergebnisse der Berechnungen übertragen werden. Bei jeder Änderung des am Bus liegenden Datenwortes werden die Kapazitäten umgeladen, und es wird dadurch ein Stromfluss von $i_c = C_p \cdot \frac{\mathrm{d}\,u}{\mathrm{d}\,t}$ hervorgerufen. Da der Stromfluss proportional zur Größe der einzelnen Kapazitäten ist, verursachen Pegeländerungen an den Busleitungen den größten Anteil der dynamischen Stromaufnahme.

An den Versorgungspins des Prozessors kann man die Summe der einzelnen dynamischen Ströme messen. Es besteht also ein direkter Zusammenhang zwischen den Änderungen der Pegel auf den Datenleitungen und der Stromaufnahme des Prozessors. Mit Hilfe verschiedener Verfahren kann man aus der Gesamtstromaufnahme eines Prozessors die Pegel von Busleitungen zu bestimmten Zeitpunkten ermitteln. Auf diese Weise ist es möglich, geheime Informationen, wie Schlüsselmaterial, zu rekonstruieren.

2.2.3. Elektromagnetische Felder

Von einem stromdurchflossenen Leiter in einem Leiter geht ein elektromagnetisches Feld aus. Der im vorigen Abschnitt aufgezeigte Zusammenhang zwischen Zustandsänderungen im Prozessor und der Stromaufnahme ist also auch für das elektromagnetische Feld, welches vom Prozessor ausgeht, gültig. Das bietet die Möglichkeit, ohne direkten Zugang zur Schaltung,

die nötig wäre, um den Strom zu messen, einen Seitenkanalangriff durchzuführen. Da jedoch die Signalqualität im Allgemeinen schlechter ist, würde die Analyse einen deutlich höheren Aufwand erfordern. Für die Analyse der Messdaten kann man auf die gleichen Methoden wie bei der Analyse der dynamischen Stromaufnahme zurückgreifen.

Eine große Gefahr geht von diesem Seitenkanal aus, wenn man in Betracht zieht, dass ein potentieller Angreifer mit einer Richtantenne eine Messung aus der Ferne durchführt. In der Praxis stellt dieses Szenario (noch) keine relevante Bedrohung dar, weil man bei einem Angriff mit solch schlechter Signalqualität viele Messungen durchführen muss.

2.2.4. Verhalten im Fehlerfall

In elektronischen Rechnern können durch externe Einflüsse, wie Strahlung oder nicht spezifizierte Betriebsparameter und Umweltbedingungen, Rechenfehler auftreten. Ein solcher Rechenfehler kann bei kryptographischen Algorithmen Geheimnisse, wie z. B. den verwendeten Schlüssel, preisgeben. Chipkarten haben oft Sensoren zur Überwachung der Betriebsparameter eingebaut, damit ein gewolltes Herbeiführen von Rechenfehlern verhindert wird.

2.3. Arten von Seitenkanalattacken

Im Laufe der letzten Jahre wurden mehrere Arten von Seitenkanalattacken mit verschiedener Komplexität entwickelt. In diesem Abschnitt soll ein Überblick über die wichtigsten Seitenkanalattacken gegeben werden. Dabei wird im Speziellen auf die *Differential Power Analysis* (DPA) genauer eingegangen, da sie Gegenstand dieser Diplomarbeit ist. In den Arbeiten [HJMS00] und [QK02] kann man eine ausführlichere Übersicht zum Thema Seitenkanalattacken finden.

2.3.1. Timing Analysis

Timing Analysis (TA) nutzt die Abhängigkeit der Rechenzeit von den zugrunde liegenden Daten aus. Im Jahr 1996 hat Paul Kocher in [Koc96] einen Angriff auf asymmetrische Verschlüsselungsverfahren vorgestellt. Dabei wurde gezeigt, dass es durch mehrmaliges Ausführen einer kryptographischen Operation und dem Messen der Ausführungszeit möglich ist, den geheimen Exponenten einer modularen Exponentiation zu ermitteln. Der zunächst hypothetische Angriff von Kocher wurde im Laufe der Zeit weiterentwickelt und konnte nicht nur auf kleinen Rechnern, wie z. B. Chipkarten, sondern auch auf komplexeren Systemen, wie PCs, durchgeführt werden. Dieser Angriff war möglich, obwohl auf einem Rechner mit einem Multitasking-Betriebssystem die Rechenzeit von der Auslastung des Rechners abhängig ist.

Im Jahr 1999 haben Dan Boneh et al. eine Timing-Attacke [BB03] auf die RSA-Exponentiation von OpenSSL vorgestellt. Bei diesem Angriff handelt es ich um um die praktische Umsetzung einer Verbesserung der ursprünglichen Timing-Attacke von Kocher. Die Theorie zu dem Angriff lieferte Schindler in der Arbeit [Sch00]. Im Gegensatz zur Timing-Attacke von Kocher ist mit dieser Verbesserung ein Angriff auf eine Implementierungen möglich, die das Chinese-Remainder-Theorem (CRT) und die Montgomery-Multiplikation verwenden. OpenSSL ist eine Programmbibliothek, welche die verschlüsselte Kommunikation über das Internet ermöglicht und z. B. bei der sicheren Variante des HTTP-Protokolls eingesetzt wird. Das Interessante an diesem Angriff war, dass es möglich war, den privaten Server-Schlüssel mit der

Länge von 1024 Bit innerhalb von zwei Stunden über eine LAN-Verbindung zu ermitteln. Man benötigte also keinen physischen Zugang zum Webserver, sondern nur die vorhandene Netzwerkverbindung. OpenSSL hatte zu dem Zeitpunkt eine entsprechende Gegenmaßnahme vorgesehen, die jedoch aus Gründen der Performanz in der Regel deaktiviert war. Nach dem Bekanntwerden dieses Angriffszenarios mussten unzählige Webserver gegen solche Angriffe abgesichert werden.

2.3.2. Simple Power Analysis

Bei der *Simple Power Analysis* (SPA) wird die Stromaufnahme während eines einzigen Durchlaufs des kryptographischen Algorithmus aufgezeichnet. Weil man nur eine Messung durchführt, muss die Güte der Messung sehr hoch sein. Das erfordert einen erheblichen messtechnischen Aufwand. Außerdem muss man eine genaue Kenntnis über die konkrete Implementierung des anzugreifenden Algorithmus besitzen, damit man Rechenoperationen der gemessenen Stromaufnahme zeitlich zuordnen kann. Sind diese beiden Voraussetzungen gegeben, so kann man Zwischenergebnisse, welche in die entsprechenden Operationen eingeflossen sind, ermitteln.

Die SPA eignet sich z. B. für einen Angriff auf den *Square-and-Multiply-Algorithmus*, der ein grundlegendes Exponentiationsverfahren bei der Implementierung von asymmetrischen kryptographischen Algorithmen ist. Der Square-and-Multiply-Algorithmus dient zur effizienten Berechnung einer modularen Exponentiation. Dabei wird die Exponentiation auf eine Folge von Quadrierungen und Multiplikationen zurückgeführt. Die Bits des Exponenten werden sequentiell abgearbeitet, was ein direktes Ablesen der einzelnen Bits aus der Stromaufnahmekurve ermöglicht.

2.3.3. Differential Power Analysis

Eine Weiterentwicklung der SPA ist die *Differential Power Analysis* (DPA). Sie wurde zusammen mit der SPA in [KJJ99] von Paul Kocher et al. vorgestellt. Im Gegensatz zur SPA wird bei der DPA nicht nur eine Stromkurve aufgezeichnet, sondern eine größere Anzahl von Stromkurven. Die Berechnung, die jeder Stromkurve zugrunde liegt, wird dabei mit jeweils verschiedenen, zufälligen Eingangsdaten durchgeführt. Der gewonnene Datensatz von Messkurven wird mit Hilfe statistischer Methoden ausgewertet. Die Qualität der Messung ist dabei nicht mehr von so großer Bedeutung wie bei SPA, da Störungen durch Mittelwertbildung herausgefiltert werden. Ein wesentlicher Vorteil der DPA gegenüber der TA und SPA ist, dass eine genaue Kenntnis über Details der Implementierung des Algorithmus nicht nötig ist. Für eine DPA müssen vier Grundvoraussetzungen gegeben sein (nach [Man04]):

- Man hat physischen Zugriff zum Rechner und kann die Stromaufnahme und somit indirekt die Leistungsaufnahme während der Berechnung messen.
- Der Klartext (P) oder der Chiffretext (C) muss dem Angreifer bekannt sein.
- Ein Zwischenergebnis der kryptographischen Operation muss anhand des Klartexts oder des Chiffretexts berechenbar sein und darf nur von einer kleinen Anzahl von Schlüsselbits abhängen.

- Die Stromaufnahme des zu untersuchenden Rechners muss abhängig von den verarbeiteten Daten sein.

In diesem Abschnitt sollen nun die zwei DPA-Analysemethoden, die im Rahmen dieser Arbeit für die praktischen Messungen verwendet wurden, näher betrachtet werden.

Bei der ersten Methode, vorgestellt von Kocher, werden die gemessenen Stromkurven abhängig von der Schlüsselhypothese und den Daten in zwei Mengen aufgeteilt. Mittels statistischem Test wird anschließend geprüft, ob die Einteilung in zwei Mengen und somit die Schlüsselhypothese richtig war.

Bei der zweiten Methode, der Korrelationsmethode, vergleicht der Angreifer einen Vektor von berechneten Bitzuständen zum Zeitpunkt t_b mit dem Vektor von gemessenen Stromwerten zu jedem Zeitpunkt $t_j, j = 1, \ldots, M$. Zu dem Zeitpunkt, an dem die vom Angreifer berechneten Bits im Messobjekt berechnet werden (T_b), ist der Vektor von Bitzuständen mit dem Vektor von Stromwerten korreliert. Dieser Umstand äußert sich in einem hohen Korrelationskoeffizienten.

Bei beiden Methoden ist es nötig, dass ein Zwischenergebnis des Algorithmus von einem kleinen Teil des Schlüssels abhängt, da ansonsten die Anzahl der Hypothesen zu groß wird. Aus diesem Grund muss der Schlüssel Stück für Stück rekonstruiert werden.

Methode nach Kocher

Als ersten Schritt verschlüsselt der Angreifer N zufällige Klartexte $P_i, i = 1, \ldots, N$ auf dem Messobjekt. Der verwendete Schlüssel ist unbekannt, muss jedoch konstant sein. Während der Berechnung der Verschlüsselungsoperationen zeichnet er zeitdiskrete Stromverbrauchskurven I_{ij} auf. I_{ij} bezeichnet den Stromverbrauch bei der Verschlüsselung des Klartexts P_i zum Zeitpunkt t_j, $j = 1, \ldots, M$.

Nun wählt der Angreifer ein Zielbit b, welches vom Klartext P und einer kleinen Anzahl von Schlüsselbits abhängt. Diese Schlüsselbits werden im Folgenden *Teilschlüssel* k_n genannt, dabei steht n für die Anzahl der Bits. Wichtig ist, dass man das Zielbit b bei bekanntem Klartext P oder bei bekanntem Chiffretext C berechnen kann. Die Funktion zur Berechnung des Bits b wird als *Auswahlfunktion* bezeichnet:

$$b = D(P, k_n) \quad \text{oder} \quad b = D(C, k_n).$$

Die gemessenen Stromverbrauchskurven werden in zwei Klassen abhängig vom angegriffenen Bit b eingeteilt

$$I^0 := \{I_{ij} | b = 0\} \quad \text{und} \quad I^1 := \{I_{ij} | b = 1\}.$$

Aus diesen beiden Klassen werden jeweils die Mittelwerte gebildet und deren Differenz berechnet

$$\bar{I}_j^0 = \left\{ \frac{1}{N_0} \sum_{i=1}^{N_0} I_{ij} | b = 0 \right\}, N_0 = \#\{I_i. | b = 0\},$$

$$\bar{I}_j^1 = \left\{ \frac{1}{N_1} \sum_{i=1}^{N_1} I_{ij} | b = 1 \right\}, N_1 = \#\{I_i. | b = 1\},$$

$$\Delta_j = \bar{I}_j^0 - \bar{I}_j^1.$$

Die Einteilung in zwei Klassen, die Mittelwertbildung und die Differenzbildung werden für jeden möglichen Teilschlüssel k_n wiederholt. Somit erhält man 2^n Differenzkurven Δ. Die Differenzkurven repräsentieren die Ergebnisse eines statistischen Tests, der nun im Folgenden genauer erklärt werden soll.

Die zwei Klassen der Stromverbrauchskurven können als Zufallsvektoren $(X_1, \ldots, X_{N_0})$ bzw. $(Y_1, \ldots, Y_{N_1})$ aufgefasst werden. Die einzelnen Zufallsvariablen X_i und Y_i sind jeweils i.i.d. (independent and indentically distributed) und entstammen den beiden unbekannten Wahrscheinlichkeitsverteilungen F_X und F_Y

$$X_i \sim F_X \quad \text{und} \quad Y_i \sim F_Y.$$

Die gemessenen Stromverbrauchskurven können dann als die Realisierung eines Zufallsexperimentes betrachtet werden, das die Werte x_i und y_i als Ergebnis hat

$$x_i \in \mathbb{R}^M, i = 1, \ldots, N_0 \quad \text{und} \quad y_i \in \mathbb{R}^M, i = 1, \ldots, N_1.$$

Nun gibt es zwei Möglichkeiten, mittels statistischer Methoden zu prüfen, ob die Einteilung in zwei Klassen richtig war. Die erste Methode ist das Prüfen mit einem nichtparametrischen Test, d. h. es werden keine Verteilungsannahmen für die Verteilungen F_X und F_Y gemacht. Die aufgestellte Nullhypothese lautet:

$$H_0 : F_X = F_Y.$$

Die entsprechende Alternativhypothese ist $H_1 : \exists x : \quad F_X(x) \neq F_Y(x)$.

Falls H_0 nicht abgelehnt wird, war die Schlüsselhypothese mit großer Wahrscheinlichkeit falsch. Falls H_0 mit der Irrtumswahrscheinlichkeit α abgelehnt wird, so stammen die Messkurven von unterschiedlichen Verteilungen und der Schlüssel wurde somit gefunden. Mögliche nichtparametrische Testverfahren sind der Kolmogorov-Smirnov-Test oder der Wilcoxon-Rank-Sum-Test.

Die zweite Möglichkeit besteht darin, einen parametrischen Test zu verwenden. Dabei wird für F_X und F_Y eine Verteilungsannahme getroffen. In unserem Fall wird angenommen, F_X und F_Y seien Normalverteilungen mit gleichen unbekannten Varianzen $\sigma^2 = \sigma_X^2 = \sigma_Y^2$ und unterschiedlichen Mittelwerten μ_X, μ_Y

$$X_i \sim \mathrm{N}(\mu_X, \sigma_X^2) \quad \text{und} \quad Y_i \sim \mathrm{N}(\mu_Y, \sigma_Y^2).$$

Die Nullhypothese für diesen Fall lautet

$$H_0 : \mu_X = \mu_Y.$$

Die Alternativhypothese ist $H_1 : \mu_X \neq \mu_Y$. Diese Hypothese kann man mit dem doppelten t-Test überprüfen. Die Teststatistik des t-Tests lautet

$$T = \frac{\bar{X} - \bar{Y}}{S_P} \sqrt{\frac{N_0 N_1}{N_0 + N_1}}$$

mit

$$S_P^2 = \frac{1}{N_0 + N_1 - 2}((N_0 - 1)S_x^2 + (N_1 - 1)S_y^2),$$
$$S_x^2 = \frac{1}{N_0 - 1}\sum_{i=1}^{N_0}(x_i - \bar{X})^2,$$
$$S_y^2 = \frac{1}{N_1 - 1}\sum_{i=1}^{N_1}(y_i - \bar{Y})^2.$$

Bei Gültigkeit von H_0 gilt $T \sim t_{N_0+N_1-2}$, d. h. die Teststatistik ist t-verteilt mit $N_0 + N_1 - 2$ Freiheitsgraden. Sei $t_{N,\alpha}$ das α-Quantil der t-Verteilung mit N Freiheitsgraden. Die Nullhypothese für den doppelten t-Test wird abgelehnt, falls

$$|T| \geq t_{N_0+N_1-2,1-\frac{\alpha}{2}}.$$

Für die Analyse nach Kocher werden S_P, N_0 und N_1 in der Teststatistik T vernachlässigt, da es sich bei einer Analyse unter den Testvoraussetzungen gleicher Varianzen um Konstanten handelt. Somit erhält man die oben beschriebene Differenzbildung der Mittelwerte. Die Differenzkurve mit dem höchsten Spitzenwert entstammt mit hoher Wahrscheinlichkeit der richtigen Schlüsselhypothese. Da man mit dieser Methode lediglich einen kleinen Teil des Schlüssels rekonstruieren kann, muss man das Verfahren mehrmals hintereinander durchführen, um den gesamten Schlüssel einer kryptographischen Operation zu finden. Dabei ist es nicht nötig die Messungen zu wiederholen, sondern man muss lediglich die Messkurven abhängig von den anderen Teilschlüsseln k_n in jeweils zwei neue Mengen einteilen und den t-Test durchführen.

Korrelationsmethode

Eine Analysemethode, welche den Korrelationskoeffizienten als Kriterium für die Richtigkeit einer Schlüsselhypothese heranzieht, wird z. B. in [Man02] verwendet.

Wie bei der Methode nach Kocher werden N Stromverbrauchskurven $I_{ij}, i = 1, \ldots, N, j = 1, \ldots, M$, aufgezeichnet. Ebenfalls wird das Bit b zum Zeitpunkt t_b abhängig vom Teilschlüssel k_n berechnet. Die berechneten Bitwerte werden in der Matrix $B_{ki}, k = 1, \ldots, 2^n, i = 1, \ldots, N$ abgelegt. Dabei bezeichnet k die Schlüsselhypothese und i die Messung.

Der Unterschied zur Methode nach Kocher besteht nun darin, dass nicht der t-Test zur Prüfung der Richtigkeit der Schlüsselhypothese herangezogen wird. Als Entscheidungskriterium für den Test kommt der Pearsonsche Korrelationskoeffizient zum Einsatz:

$$\rho_{kj}(B_{ki}, I_{ij}) = \frac{\mathrm{E}(B_{ki} \cdot I_{ij}) - \mathrm{E}(B_{ki}) \cdot \mathrm{E}(I_{ij})}{\sqrt{\mathrm{Var}(B_{ki}) \cdot \mathrm{Var}(I_{ij})}}.$$

Als Ergebnis bekommt man eine Matrix von Korrelationskoeffizienten $\rho_{kj}, k = 1, \ldots, 2^n, j = 1, \ldots, M$, in der jede Zeile für eine Schlüsselhypothese und jede Spalte für einen Zeitpunkt t_j steht.

Wenn man jede Zeile der Matrix als Korrelationskurve für eine bestimmte Schlüsselhypothese auffasst, hat man ein den Differenzkurven der Methode nach Kocher „äquivalentes“ Ergebnis. Zusätzlich besteht der Vorteil, dass die Kurven auf ± 1 normiert sind. Das Ergebnis

ist analog zum Ergebnis nach Kocher zu interpretieren, d. h. die Kurve mit der größten Spitze geht mit großer Wahrscheinlichkeit aus der richtigen Schlüsselhypothese hervor.

Die Korrelationsmethode funktioniert deshalb, weil, wie in Abschnitt 3.2.2 gezeigt wird, ein direkter Zusammenhang zwischen Stromaufnahme und berechneten Bit besteht.

Da, wie bei der Methode nach Kocher, in jedem Durchgang der Analyse nur ein kleiner Teil des Schlüssels rekonstruiert werden kann, muss die Analyse der gemessenen Stromkurven für jeden weiteren Teilschlüssel wiederholt werden.

Auswahlfunktion des DES

In diesem Abschnitt soll die standardmäßige Auswahlfunktion für den *Data Encryption Standard* DES ([NIS99], [MvOV96, S. 250ff]) vorgestellt werden. Der Algorithmus hat eine Blocklänge von 64 Bit und eine Schlüssellänge von 56 Bit. Da die Größe des Schlüssels mit heutiger Rechenleistung eine Brute-Force-Attacke ermöglicht, wird der Algorithmus heutzutage in der Regel dreimal hintereinander ausgeführt, um den Schlüsselraum zu vergrößern (Triple-DES). Die zu verschlüsselnden Daten durchlaufen beim DES 16 Runden. Die Rundenfunktion hat den Aufbau

$$\mathrm{f}(L, R, k) = L \oplus \mathrm{P}(\mathrm{S}(\mathrm{E}(R) \oplus k)).$$

Dabei bezeichnet ...

- E() die Expansion (32 Bit → 48 Bit), bei der einzelne Bits verdoppelt werden.
- L, R die linke und rechte Hälfte, in die der Datenstrom aufgeteilt wird (jeweils 32 Bit).
- k den Rundenschlüssel, der vom 56 Bit langen DES–Schlüssel abgeleitet wird (48 Bit).
- P() eine definierte Permutation der Bits.
- S() die acht S-Boxen, welche eine nichtlineare Abbildung darstellen.

Abbildung 2.2 zeigt den Ausschnitt der DES-Rundenfunktion, der als Auswahlfunktion dient. Die finale XOR-Verknüpfung könnte entfallen, wird jedoch miteinbezogen, da so das angegriffene Bit in die zweite Runde weitergereicht wird und in mehrere Rechenoperationen einfließt, was die DPA-Analyse erleichtert. Bei der dargestellten Auswahlfunktion wird die erste Runde des DES bei bekanntem Klartext P angegriffen. Eine andere Variante der Auswahlfunktion ergibt sich aus der letzten Runde bei bekanntem Chiffretext C.

Der Teilschlüssel, welcher in die Auswahlfunktion einfließt, hat eine Länge von $n = 6$ Bit, daraus resultieren $2^6 = 64$ Schlüsselhypothesen. Eine Runde besitzt acht S-Boxen, d. h. man kann mit der Analyse einer Runde 48 Bits des Schlüssels rekonstruieren.

Beim DES ist es somit möglich, durch einmalige Durchführung der DPA-Messung, achtfacher Auswertung für jede S-Box und anschließender Brute-Force-Suche für die fehlenden acht Bit den kompletten Schlüssel zu ermitteln. Beim Three Key Triple DES verdreifacht sich ungefähr der Aufwand.

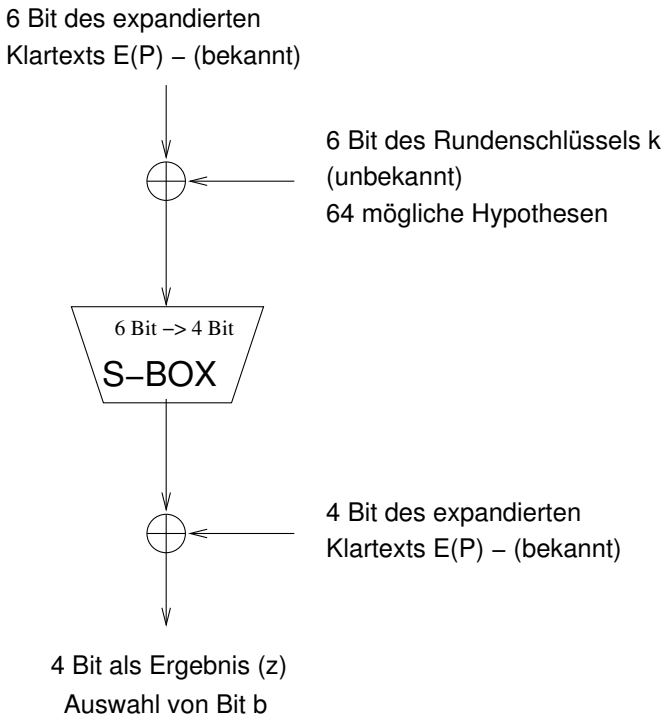

Abbildung 2.2: Eine einfache DES-Auswahlfunktion $b = \mathrm{D}(P, k_6)$

Auswahlfunktion des AES

In diesem Abschnitt soll die standardmäßige Auswahlfunktion für den Advanced Encryption Standard AES ([NIS01], [DR99]) vorgestellt werden. Beim AES handelt es sich um den Nachfolger des DES, der nötig wurde, da der DES für heutige Verhältnisse eine zu kleine Schlüssellänge und eine zu kleine Blockgröße aufweist. Der AES wurde in Form eines öffentlichen Auswahlprozesses aus vorgeschlagenen Algorithmen ausgewählt. An die AES-Kandidaten wurden verschiedene Anforderungen, wie z. B. die Möglichkeit von effizienten Implementierungen in Software sowie in Hardware, gestellt.

Analog zur Auswahlfunktion des DES soll nun die erste Runde bei bekanntem Klartext P analysiert werden. Wie beim DES könnte man auch die letzte Runde mit bekanntem Chiffretext C angreifen. Der AES hat eine stark byteorientierte Struktur, was ihn für Implementierungen auf verschiedenen Prozessoren besonders schnell macht. Die Blocklänge des AES beträgt 128 Bit, die Schlüssellänge ist variabel 128, 196 oder 256 Bit lang. Die Anzahl der Runden ist abhängig von der Schlüssellänge. Wir wollen uns in dieser Arbeit auf eine Schlüssellänge von 128 Bit und somit zehn Runden beschränken. Eine Runde des AES besteht aus vier verschiedenen Transformationen wie dieser C-Pseudocode zeigt:

Listing 2.2: Rundenfunktion des AES-Algorithmus

```
AES_Round(State, RoundKey) {
    ByteSub(State);
    ShiftRow(State);
    MixColumn(State);
    AddRoundKey(State, RoundKey);
}
```

Die Variable `State` enthält das jeweilige Zwischenergebnis und `RoundKey` den entsprechenden Rundenschlüssel. Vor der ersten Runde erfolgt eine XOR-Verknüpfung von Klartext und Rundenschlüssel k_0. Der Rundenschlüssel k_0 entspricht genau dem 128 Bit AES-Schlüssel. In der letzten Runde entfällt die Operation `MixColum()`.

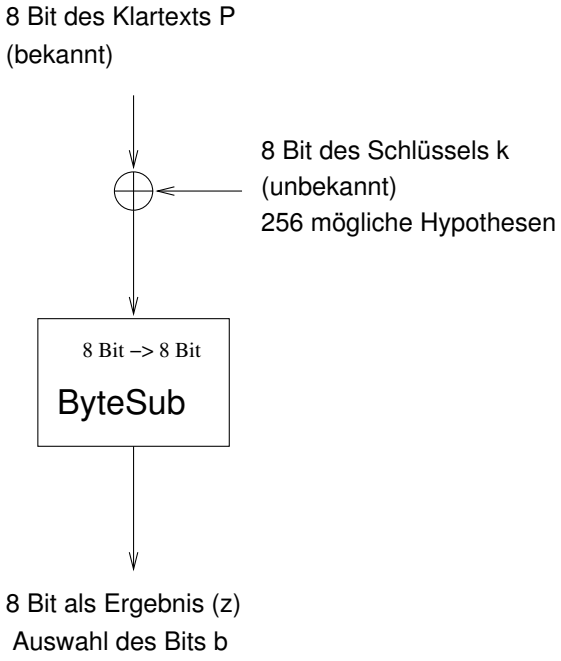

Abbildung 2.3: Eine einfache AES-Auswahlfunktion $b = \mathrm{D}(P, k_8)$

Wie in Abbildung 2.3 gezeigt, wird die XOR-Operation vor der ersten Runde und die nichtlineare `ByteSub()`-Operation der ersten Runde vom Angreifer nachgebildet. Wegen der Byteorientierung ist der kleinste Teilschlüssel, der in eine Berechnung einfließt, acht Bit lang. Man muss also $2^8 = 256$ Schlüsselhypothesen aufstellen, was einen größeren Speicherbedarf gegenüber DES zur Folge hat.

Durch sequentielles Abarbeiten aller 16 Bytes kommt man zu allen 128 Bit des Schlüssels. Man kann also alle Schlüsselbits mit der Attacke auf eine Runde ohne die Zuhilfenahme einer Brute-Force-Attacke, wie es beim DES nötig ist, rekonstruieren.

Erweiterung der Auswahlfunktion – Multiple-Bit-DPA

Ist das DPA-Signal so stark verrauscht, dass die Stromdifferenz eines einzigen Bits trotz Mittelwertbildung nicht mehr sichtbar ist, dann besteht trotzdem noch eine Möglichkeit, eine

DPA-Analyse durchzuführen. Dabei werden die Auswahlfunktionen, wie sie in den letzten zwei Abschnitten vorgestellt wurden, so modifiziert, dass nicht ein Bit, sondern alle Bits des Zwischenergebnisses z ausgegeben werden

$$z = D'(P, k_n) \quad \text{oder} \quad z = D'(C, k_n).$$

Die Größe z ist beim DES vier und beim AES acht Bits lang. Die Kurven werden nicht mehr, wie bei der Single-Bit-DPA in zwei Mengen eingeteilt, sondern in drei. Die Einteilung erfolgt abhängig von der unteren Schranke u und der oberen Schranke o des Hamminggewichts[1] hw()

$$I^0 := \{I_{ij} | \, \mathrm{hw}(z) < u\} \quad \text{und} \quad I^1 := \{I_{ij} | \, \mathrm{hw}(z) > o\} \quad \text{und} \quad I^2 := \{I_{ij} | u \leq \mathrm{hw}(z) \leq o\}.$$

Die dritte Menge I^2 enthält Kurven, die nicht in die Berechnung einfließen, also verworfen werden. Mit den ersten zwei Mengen (I^0 und I^1) wird die Analyse analog zur Single-Bit-DPA durchgeführt. Bei einem Angriff auf den DES-Algorithmus mit der Einstellung $u = 1$ und $o = 3$ werden alle Kurven bis auf die mit $z = 0$ und $z = 15$ verworfen. Das erfordert eine ungefähr achtfache Anzahl von Messungen gegenüber der Single-Bit-DPA. Da bei dieser Diplomarbeit das Verbessern der Signalqualität durch entsprechende Hardwaremodifikationen im Vordergrund steht, wurde diese Form der Analyse nicht angewendet.

2.3.4. SEMA / DEMA

Im Prinzip sind *Simple Electromagnetic Analysis* (SEMA) und *Differential Electromagnetic Analysis* (DEMA) zur SPA und DPA äquivalent. Im einfachsten Fall kann man die gemessenen Daten mit den gleichen Methoden wie bei der SPA/DPA auswerten. In der Regel ist jedoch das Signal-Rausch-Verhältnis der Messdaten schlechter als bei der einfachen Strommessung.

Der Vorteil der Analyse des elektromagnetischen Feldes besteht darin, dass man das Messobjekt nicht durch einen Shunt-Widerstand erweitern muss. Die Messung kann an einem beliebigen Ort innerhalb des Feldes erfolgen. Daraus ergibt sich die Möglichkeit, dass man Messungen in größerer Entfernung zum Messobjekt (Fernfeld) oder sehr nahe am Chip (Nahfeld) vornehmen kann. Die Messung könnte an verschiedenen Stellen zugleich durchgeführt werden und die gewonnen Daten könnten kombiniert ausgewertet werden. Dadurch könnte ein Angreifer mit geeigneten Auswertemethoden mehr Information gewinnen als aus einer einzigen Datenquelle.

Ein großes Potential besteht bei der Messung sehr nahe am Chip. Dabei könnte der Angreifer gezielt einen bestimmten Schaltungsteil „abhören“ und so das Signal-Rausch-Verhältnis verbessern.

2.3.5. Fault Attacks

Wie schon eingangs erwähnt, können Rechenfehler, die bei der Ausführung eines kryptographischen Algorithmus auftreten, geheime Informationen preisgeben. Ein extremes Beispiel ist die Anfälligkeit des Chinese-Remainder-Theorem (CRT) [BDL97], das in praktischen Implementierungen von RSA (siehe [MvOV96, Kapitel 8.2]) zum Einsatz kommt, um die Berechnung

[1] Anzahl der gesetzten Bits

der Exponentiation zu beschleunigen. Die Signatur einer Nachricht m mit dem geheimen Schlüssel d erfolgt mittels RSA folgendermaßen

$$s \equiv m^d \bmod n \text{ mit } n = p \cdot q.$$

Mit Hilfe des CRT kann der Rechenaufwand um ein Viertel reduziert werden, dazu muss jedoch p und q bekannt sein und geheim gehalten werden:

$$\begin{aligned} s_1 &\equiv m^{d \bmod p-1} \bmod p, \\ s_2 &\equiv m^{d \bmod q-1} \bmod q, \\ s &\equiv a \cdot s_1 + b \cdot s_2 \bmod n. \end{aligned}$$

Die Werte a und b sind konstant und können aus p und q vorausberechnet werden.

Tritt bei der Berechnung der Signatur ein Fehler auf, so tritt der Fehler mit großer Wahrscheinlichkeit bei der Berechnung von s_1 oder s_2 auf. Die Berechnung von s_1 und s_2 verursacht nämlich fast die gesamte Rechenzeit. Wichtig ist, dass der Fehler nur bei einer der beiden Größen auftritt. Die resultierende fehlerhafte Signatur wird als s' bezeichnet. Gelten die Beziehungen $s \equiv s' \bmod q$ und $s \not\equiv s' \bmod p$, so kann ein potentieller Angreifer q berechnen und so n faktorisieren

$$q = \mathrm{ggT}(s - s', n).$$

Damit kann der Angreifer den geheimen Schlüssel sehr einfach bestimmen.

2.4. Maßnahmen gegen Seitenkanalattacken

Maßnahmen gegen Seitenkanalattacken kann man durch Änderungen an der Hardware oder durch Änderungen an der Software implementieren. Eine grundlegende Gegenmaßnahme ist dabei, interne Rechenschritte und die von außen messbaren Größen, wie Stromaufnahme oder Rechenzeit, zu dekorrelieren. Oft wird auch einfach das Signal-Rausch-Verhältnis durch künstlich erzeugtes Rauschen verschlechtert. Das löst aber nicht das prinzipielle Problem der Anfälligkeit gegen Seitenkanalattacken. Es wird lediglich der Aufwand für die Analyse erhöht.

Im Laufe der Zeit wurden verschiedene Maßnahmen für diverse Anwendungsfälle entwickelt, die mehr oder weniger wirkungsvoll sind. In diesem Abschnitt soll ein Überblick über mögliche Gegenmaßnahmen gegeben werden. Die Entwicklung effizienter Gegenmaßnahmen ist ein aktives Forschungsgebiet. Viele Maßnahmen sind patentrechtlich geschützt oder nicht veröffentlicht. Detailierte Informationen über veröffentliche Gegenmaßnahmen bieten die Tagungsbände der CHES-Konferenzen [KP99, KP00, KNP01, KKP02, WKP03, JQ04].

2.4.1. Software-Gegenmaßnahmen

Gegen Timing Attacken kann man einen Algorithmus so modifizieren, dass die Rechenzeit unabhängig von den Daten wird. Dabei geht jedoch Rechenzeit verloren, da jede Berechnung so lange wie die langsamste dauern muss. Deshalb ist diese Form der Gegenmaßnahme nicht immer praktikabel. Das Einfügen von zufälligen Wartezeiten kann den Angriff zwar erschweren, jedoch nicht prinzipiell verhindern.

Maßnahmen gegen die Analyse von fehlerhaften Berechnungen bauen darauf auf, dass der Algorithmus intern überprüft, ob ein Fehler aufgetreten ist. Das Überprüfen kann mittels

Checksummen, mehrmaligem Durchführen der Berechnung und Vergleich der Ergebnisse oder durch eine zusätzliche inverse Berechnung, nach der der Ausgangswert mit dem Ergebnis der inversen Berechnung verglichen wird, erfolgen. Wird ein Fehler entdeckt, darf das Ergebnis nicht ausgegeben werden.

Eine viel versprechende Methode gegen statistische Angriffe ist das Randomisieren der Daten. Dabei wird vor der Berechnung des Algorithmus der Eingangswert mit einer Zufallszahl verknüpft, die nach der Ausführung des Algorithmus wieder aus dem Ergebnis heraus gerechnet werden muss. Dadurch ist es dem Angreifer unmöglich, ein Zwischenergebnis des Algorithmus zu berechnen. Dieses Verfahren wird als Maskierungstechnik bezeichnet. In [AG01] werden Verfahren vorgestellt, die es ermöglichen, DES und AES zu maskieren, so dass es einen Angreifer nicht möglich ist, ein Zwischenergebnis zu berechnen. Somit ist eine einfache DPA-Analyse nicht möglich. In dieser Diplomarbeit wird eine DES-Implementierung, welche auf diesem Aufsatz basiert, mittels einfacher DPA-Analyse angegriffen. Der Angriff hat aus dem eben genannten Grund nicht funktioniert.

2.4.2. Hardware-Gegenmaßnahmen

Ein sehr effizienter Schutz kann mit der Kombination von Software- und Hardware-Gegenmaßnahmen erreicht werden. Eine einfache, jedoch nicht sehr wirkungsvolle Hardware-Maßnahme gegen die Analyse der Stromaufnahme ist das Verrauschen das Signals. Diese Maßnahme erhöht nur den Aufwand bei Messung und Auswertung der Seitenkanalinformation. Ideal wäre es, wenn die Stromaufnahme unabhängig vom internen Zustand der Schaltung ist. Ein Ansatz dafür ist eine so genannte Dual-Rail-Logik, bei der zwei Busleitungen ein Bit übertragen. Weitere denkbare Maßnahmen könnten eine variable Ausführungszeit von Befehlen, ein intern erzeugter Takt, der nicht konstant ist und asynchrone Schaltungstechnik sein.

Zertifizierung Seitenkanalresistenz ist für die Zertifizierung eines Algorithmus nach ITSEC (Information Technology Security Evaluation Criteria) oder CC (Common Criteria) sehr bedeutend. Gegen einfache Angriffe, wie TA, SPA und DPA, ist die Seitenkanalresistenz bei hohem Angriffspotential zwingend vorgeschrieben. Mächtigere Angriffsmethoden, wie SEMA, DEMA oder Higher Order DPA, werden bei der Zertifizierung noch nicht berücksichtigt.

Kapitel 3.

Praktische Umsetzung

3.1. Laborarbeitsplatz

Der Laborarbeitsplatz zur Durchführung der Seitenkanalanalysen befindet sich in der Abteilung Corporate Technology, Information and Communication, Security Technologies, (CT IC 3) der Siemens AG. In Abbildung 3.1 sind die wichtigsten Komponenten des Arbeitsplatzes zu sehen. Es handelt sich dabei um einen *leistungsfähigen PC* zur Auswertung der Messdaten, um ein *digitales Speicheroszilloskop* und um *verschiedene Messobjekte*, die einer Analyse unterzogen werden.

Abbildung 3.2 zeigt ein Blockschaltbild, das das Zusammenspiel der einzelnen Komponenten illustrieren soll. Das Messobjekt wird über eine serielle Schnittstelle (RS232) vom Laborrechner angesteuert. Über diese Schnittstelle ist es möglich, den kryptographischen Algorithmus auf dem Messobjekt zu starten. Das Messobjekt erzeugt ein Triggersignal, wenn es zu rechnen beginnt. Das Triggersignal startet die Strommessung durch das Oszilloskop. Die Stromaufnahme des Messobjekts wird indirekt als Spannungsabfall an einem Shunt-Widerstand vom Oszilloskop aufgezeichnet. Die aufgenommene Stromkurve wird über die GPIB-Schnittstelle, einem speziellen Labor-Bus, zum Laborrechner übertragen. Die Ansteuerung des Messobjekts und die Auswertung der Messdaten erfolgten innerhalb des Mathematikprogramms Matlab[1].

In diesem Abschnitt soll nun ein kurzer Überblick über die einzelnen verwendeten Komponenten gegeben werden.

3.1.1. Laborrechner

Der Laborrechner ist ein handelsüblicher, jedoch etwas leistungsfähigerer Intel-PC. Er verfügt über eine große Festplatte, um die umfangreichen Messdaten speichern zu können und eine spezielle GPIB-Schnittstelle, über die man Laborgeräte, wie zum Beispiel ein Speicheroszilloskop, ansteuern kann. In Tabelle 3.1 sind die wichtigsten Kenndaten dieses PCs zusammengefasst.

3.1.2. Oszilloskop und Tastköpfe

Bei dem verwendeten Oszilloskop handelt es sich um ein WavePro 7200 von LeCroy[2], ein sehr leistungsfähiges digitales Speicheroszilloskop mit 2 GHz analoger Bandbreite und vier Kanälen. Es basiert auf einem Industrie-PC, welcher unter Microsoft Windows 2000 Embedded betrieben wird. Wichtig für unsere Anwendung ist die Schnittstelle, über die man die Messdaten zu einem PC übertragen kann. In unserem Fall geschieht das über die GPIB-Schnittstelle,

[1] http://www.mathworks.com
[2] http://www.lecroy.com

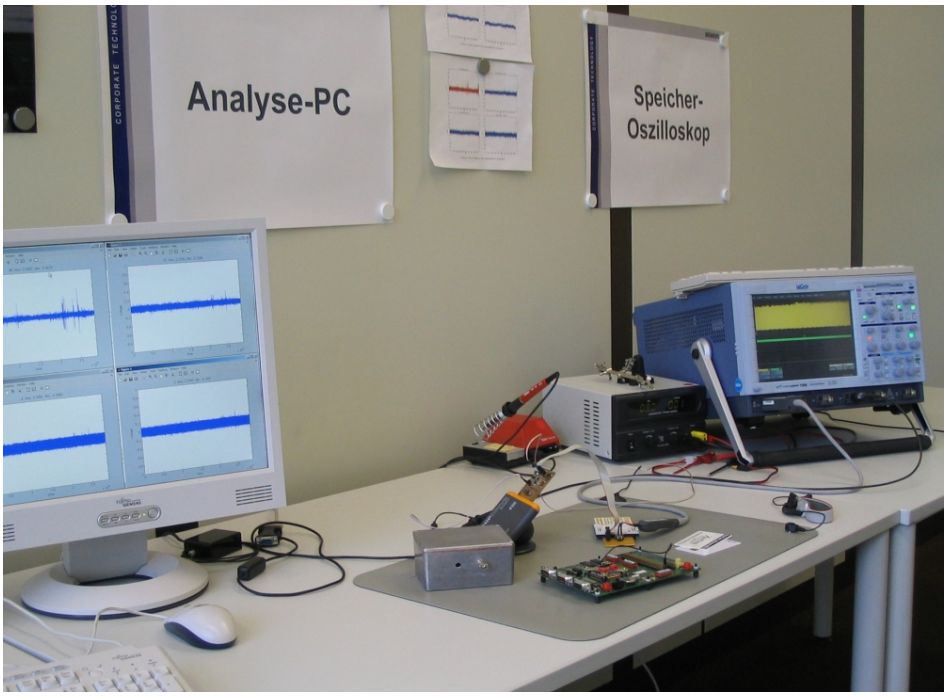

Abbildung 3.1: Foto des Laborarbeitsplatzes

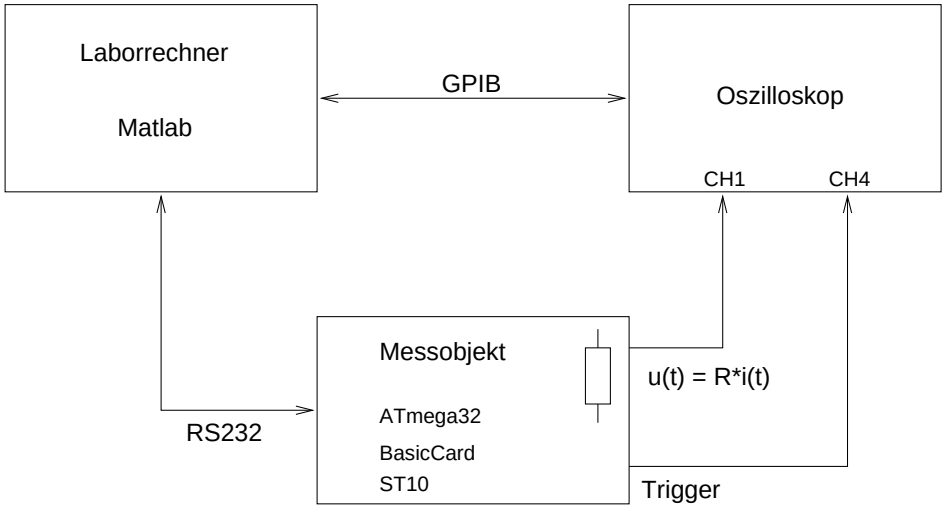

Abbildung 3.2: Blockschaltbild des Laborarbeitsplatzes

Tabelle 3.1: Konfiguration des Laborrechners

Komponente	Beschreibung
Prozessor	Intel Pentium 4 / 3.2 GHz
RAM-Ausstattung	4 GB
Festplatten	400 GB und 160 GB SATA
Smart Card Reader	SCM Microsystems Inc. CHIPDRIVE Serial Gemplus USB Smart Card Reader
GPIB-Karte	National Instruments PCI-GPIB NI-488.2
Betriebssystem	Microsoft Windows XP Service Pack 2 alternativ SuSE Linux 9.2
Primäre Softwareumgebung für Datenanalysen	Matlab 7.0.1 (R14) Service Pack 1 Instrument Control Toolbox 2.1.2 Neural Network Toolbox 4.0.4 Signal Processing Toolbox 6.2.1 Statistics Toolbox 5.0.1
Compiler	Microsoft Visual Studio .Net 2003 Tasking C166-ST10 v8.0/v8.5 Keil μVision2/3 mit C-Compiler Version 5.0

ein Bussystem, welches für die Kommunikation zwischen Laborgeräten konzipiert wurde. Das Oszilloskop verfügt alternativ über eine Ethernetschnittstelle, die zur Zeit nicht genutzt wird. Tabelle 3.2 gibt einen Überblick über die wichtigsten Kenndaten des Oszilloskops.

Tabelle 3.2: Kenndaten des Oszilloskops

Typenbezeichnung	LeCroy WavePro 7200
Kommunikationsschnittstellen	GPIB und Ethernet
Speichertiefe (Messdaten)	16 MB
Auflösung	8 Bit
Maximale Abtastrate	20 $^{\mathrm{GS}}/_{\mathrm{s}}$
Analoge Bandbreite	2 GHz
Eingangsimpedanz	50 Ω oder 1 MΩ

Die Messgrößen werden mittels Tastkopf abgegriffen. Der Tastkopf soll eine Verbindung zwischen dem Messobjekt und dem Oszilloskop herstellen und dabei das Messergebnis möglichst wenig verfälschen. Grob unterscheidet man zwischen aktiven und passiven Tastköpfen, mit dem Unterschied, dass ein aktiver Tastkopf zusätzlich mit einem breitbandigen Verstärker ausgestattet ist. Dadurch erreicht man eine größere Messgenauigkeit gegenüber einem passiven Tastkopf. Für die DPA-Messungen wurde ein aktiver differentieller Tastkopf verwendet. Differentiell bedeutet, dass dieser Tastkopf nicht die Masse als Bezugspotenzial verwendet, sondern eine Spannungsdifferenz misst. Eine solche Spannungsdifferenz tritt z. B. an einem stromdurchflossenen Widerstand auf. Für das Abgreifen des Triggersignals wurde ein passiver Tastkopf verwendet, da es beim Triggersignal nicht auf hohe Signalqualität ankommt. Aus

Tabelle 3.3 kann man die Kenndaten der verwendeten Tastköpfe entnehmen. Abbildung 3.3 zeigt die Ersatzschaltungen der Tastköpfe.

Tabelle 3.3: Kenndaten der verwendeten Tastköpfe

Typenbezeichnung	LeCroy AP034	LeCroy PP005
Art des Tastkopfes	aktiv, differentiell	passiv
Analoge Bandbreite	1 GHz	500 MHz
Dämpfung	1:1	1:10
Eingangskapazität	0,85 pF	11 pF
Eingangswiderstand	2 MΩ	10 MΩ

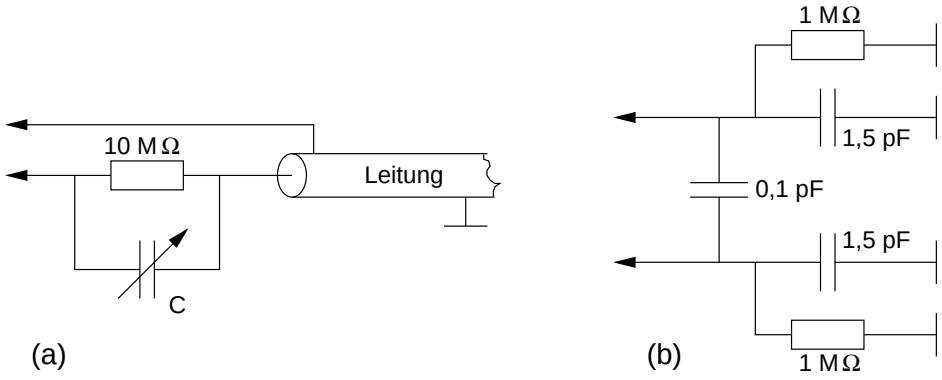

Abbildung 3.3: Ersatzschaltbilder des passiven (a) und des aktiven differentiellen (b) Tastkopfes

3.1.3. Messobjekt

Im Seitenkanallabor hat man die Möglichkeit, verschiedene kryptographische Rechner zu testen. Man kann Softwareimplementierungen auf Mikrocontrollern auf ihre Resistenz gegen DPA-Attacken prüfen, bevor sie als Embedded-System zum Einsatz kommen.

Im Rahmen dieser Diplomarbeit wurden zwei verschiedene Mikrocontroller verwendet. Beim Ersten handelt es sich um den 8 Bit RISC-Controller ATmega32. Der zweite Mikrocontroller ist der 16 Bit CISC-Controller ST10F168, der vorwiegend in der Automobilindustrie zum Einsatz kommt. Beide Mikrocontroller werden über die serielle Schnittstelle mit dem PC verbunden. Die Schnittstelle wird mit 9600 Baud und der Einstellung 8N1[3] betrieben. Der aufgenommene Strom wird als Spannungsabfall an einem Shunt-Widerstand gemessen. Für den ATmega32 wurde im Rahmen der Diplomarbeit eine eigene Platine entwickelt, auf der sich der Shunt-Widerstand und eine entsprechende Anschlussmöglichkeit für das Oszilloskop befinden. Der ST10168 befindet sich auf einem Evaluationsboard. Das Board musste erweitert werden, damit man die Strommessung direkt am Mikrocontroller vornehmen kann. Die Triggerung des Oszilloskops erfolgt bei beiden Mikrocontrollern über einen I/O-Pin, der von der Software im Controller angesteuert wird.

[3] 8 Datenbits, keine Parität, 1 Stopbit

Für das Analysieren von Chipkarten wurde ein spezieller Adapter gebaut. Damit ist es möglich, Messungen ohne das Verändern des Chipkartenlesers oder der Chipkarte vorzunehmen. Die Erzeugung des Triggersignals ist bei Chipkarten nicht direkt möglich, weil die Chipkarte keine frei programmierbaren I/O-Pins besitzt. Deshalb wurde eine Mikrocontrollerschaltung entwickelt, die das Chipkartenkommunikationsprotokoll überwacht und bei Empfang eines Verschlüsselungskommandos das Oszilloskop triggert.

3.2. Messtechnik und EMV

Messfehler kann man grob in systematische und zufällige Messfehler gruppieren. Systematische Messfehler haben innerhalb einer Messreihe einen konstant auftretenden Fehler zur Folge. Der Grund für systematische Fehler liegt z. B. bei defekten oder falsch justierten Messgeräten oder generellen Fehlern im Messaufbau. Zufällige Messfehler entstehen durch zufällige Einwirkungen auf einzelne Messungen. Verschiedene zufällige Messfehler kann man allgemein unter dem Begriff Rauschen zusammenfassen. Gemäß [Mes00a] kann man die Anteile am Rauschen einer DPA-Messung in fünf Teile gliedern:

- **Externes Rauschen:**
  Externes Rauschen wird durch Störquellen in Form von elektromagnetischer Strahlung hervorgerufen. Externe Störeinflüsse kann man verringern, indem man mögliche Störquellen ausschaltet bzw. räumlich trennt oder indem man den Messaufbau von Umwelteinflüssen abschirmt. Abschnitt 3.2.1 zeigt mögliche Störquellen, Koppelmechanismen und Gegenmaßnahmen auf.

- **Intrinsisches Rauschen:**
  Unter intrinsischem Rauschen versteht man das Rauschen, welches im Inneren der Schaltung durch Ladungsträgertransport entsteht. Bei Halbleitern tritt das so genannte Schrotrauschen an PN-Übergängen auf, was durch zufälligen Ladungstransport durch das Feld der Verarmungszone hervorgerufen wird.

  An Widerständen tritt thermisches Rauschen auf, was auf Ladungsträgerbewegungen innerhalb eines Widerstandes zurückzuführen ist

  $$\frac{P}{R} = i^2 = 4kT\frac{1}{R}\Delta f.$$

  Dabei ist P die Rauschleistung, i der Rauschstrom, k die Bolzmann-Konstante $k = 1.38 \cdot 10^{-23}$ J/K, T die absolute Temperatur, R der Widerstand und Δf die Frequenzbandbreite. Als Abhilfe gegen thermisches Rauschen kann man die Temperatur und die Bandbreite verringern. Ein Angreifer könnte also das zu messende Objekt kühlen und das Messsignal durch Filterung aufbereiten.

- **Quantisierungsrauschen**:
  Beim Digitalisieren werden analoge Messdaten zeit- und wertdiskretisiert. Bei der Wertdiskretisierung wird ein vorher analoger Spannungswert in eine digitale Zahl umgewandelt. Dabei geht unweigerlich Information verloren, was sich in Form von Quantisierungsrauschen äußert. Wenn man beispielsweise eine analoge Spannung von 0 V bis 5 V mit 8 Bit quantisiert, so ist der kleinste darstellbare Spannungssprung $\frac{5\,\mathrm{V}}{255} = 19,6\,\mathrm{mV}$. Man bekommt also einen mittleren Quantisierungsfehler von $9,8\,\mathrm{mV}$. Eine Abhilfe gegen Quantisierungsrauschen ist das Verwenden von höher auflösenden Analog-Digital-Wandlern.

- **Rauschen wegen Phasenjitter**:
  Bei DPA-Messungen werden viele Messkurven aufgenommen, aus denen Mittelwerte gebildet werden. Die Mittelwertbildung soll dabei Rauschen, vor allem externes, intrinsisches, algorithmisches und Quantisierungsrauschen verringern. Dabei kann jedoch durch die Mittelwertbildung zusätzliches Rauschen erzeugt werden. Sind nämlich die einzelnen Messkurven zeitlich nicht exakt ausgerichtet, werden bei der Summenbildung Messwerte, die keinen kausalen Zusammenhang haben, addiert, was das Endergebnis verfälscht.

- **Algorithmisches Rauschen**:
  Da die Stromaufnahme von den *Zustandsübergängen* einer Digitalschaltung abhängt, hat die Abarbeitung eines kryptographischen Algorithmus ein Rauschen im Messsignal zur Folge. Dieses algorithmische Rauschen hängt von der Reihenfolge der Maschinenbefehle, also von der konkreten Implementierung des Algorithmus, ab.

3.2.1. Externes Rauschen und Koppelmechanismen

Die Störunempfindlichkeit von Schaltungen ist ein Teilbereich der *Elektromagnetischen Verträglichkeit* (EMV). In diesem Abschnitt sollen die grundlegenden Mechanismen, die zu Messfehlern durch Kopplungen führen, besprochen werden. Da das Thema sehr komplex ist, kann nur oberflächlich auf die Problematik eingegangen werden. Einen leicht verständlichen Einstieg in die Thematik EMV bieten [Rod00] und [Fra02]. In diesem Abschnitt soll nicht auf physikalische Grundlagen eingegangen werden, sondern die Problematik an Hand von anschaulichen Beispielen erläutert werden. Dabei beschränken sich die Ausführungen auf quasistationäre Felder.

Einen Koppelmechanismus kann man grundlegend in drei Bereiche gliedern: Zum einen hat man eine Störquelle, welche oft unvermeidbar ein Störsignal erzeugt. Das Störsignal wird über eine Kopplung zur Störsenke transportiert, wo man eine unerwünschte Störung des Nutzsignals hervorruft. Bei der Vermeidung von Störungen muss man bei der Kopplung ansetzen und dadurch eine Übertragung von Quelle zu Senke verhindern. Abbildung 3.4 zeigt diesen grundlegenden Zusammenhang. In den folgenden Unterabschnitten sollen die vier wichtigsten Koppelmechanismen und mögliche Gegenmaßnahmen aufgezeigt werden. Die Koppelmechanismen treten meistens gemeinsam auf, was Kopplungen in komplexeren Systemen schnell unübersichtlich machen kann.

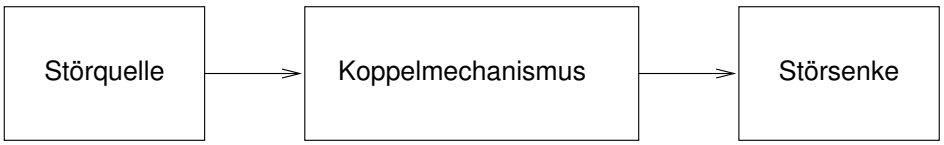

Abbildung 3.4: Schematische Darstellung des Koppelmechanismus

Induktive Kopplung

Ein von einem sich zeitlich ändernden Strom durchflossener Leiter ist stets von einem sich zeitlich ändernden Magnetfeld umgeben. Dieses Magnetfeld kann wiederum einen Stromfluss in einer benachbarten Leiterschleife hervorrufen. Dieses Phänomen nennt man *transformatorisches Prinzip* bzw. *induktive Kopplung*. Die induktive Kopplung, welche bei einem Transformator oder Übertrager gewünscht ist, kann z. B. bei einer Messschaltung höchst unerwünscht sein. Eine solche unerwünschte Kopplung kann man sich als einen Transformator mit nur einer primären und einer sekundären Wicklung vorstellen. Abbildung 3.5 zeigt den geometrischen Aufbau und das Ersatzschaltbild einer induktiven Kopplung.

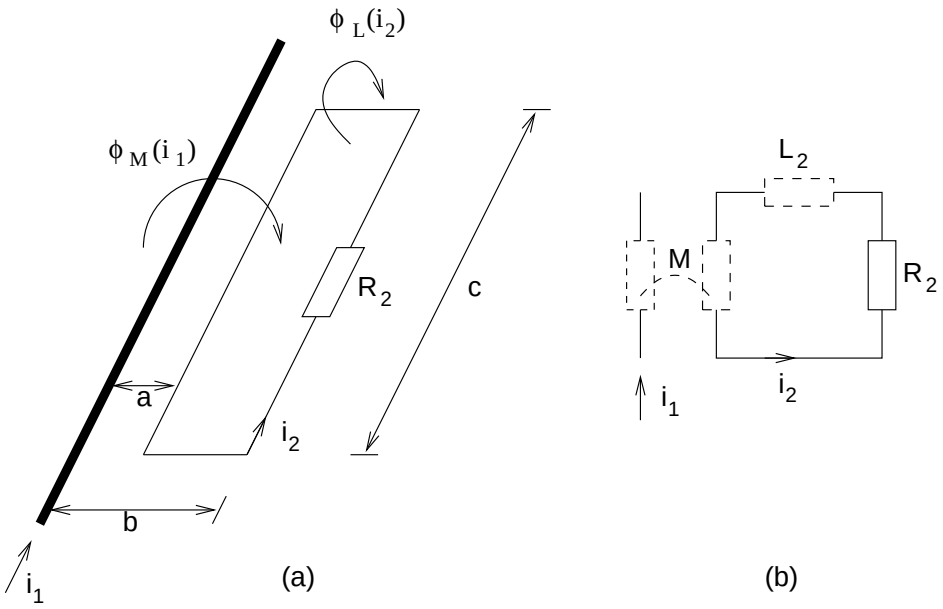

Abbildung 3.5: Induktive Kopplung, geometrische Anordnung (a) und Ersatzschaltbild (b)

Der Strom $i_1(t)$ hat den magnetischen Fluss $\phi_M(i_1)$ zur Folge, welcher in der Ersatzschaltung (siehe Abbildung 3.5 b) als Gegeninduktivität M dargestellt wird. Der Fluss $\phi_M(i_1)$ erzeugt eine transformatorische Spannung U_{TR}, welche wiederum einen Stromfluss $i_2(t)$ erzeugt. Der Strom $i_2(t)$ verursacht einen magnetischen Fluss $\phi_L(i_2)$, der in der Ersatzschaltung als Eigeninduktivität L_2 dargestellt wird. Der Widerstand R_2 bezeichnet den Innenwiderstand der Störsenke.

Der Zusammenhang zwischen Strom $i_1(t)$ und $i_2(t)$ ist folgendermaßen gegeben, wobei p

für die komplexe Kreisfrequenz $p = j \cdot \omega$ steht und $\omega = 2\pi f$ ist

$$i_2 = \frac{pM}{R_2 + pL_2} \cdot i_1.$$

Der Strom i_2 hängt also von der Frequenz p und den Bauteilwerten M, R_2 und L_2 ab. Da man in der Regel die Störsenke nicht verändern kann und die Frequenz des Störsignals ebenfalls nicht beeinflussbar ist, muss man die Gegeninduktivität M verringern.

Die Abhängigkeit der Gegeninduktivität M von der Geometrie der Leiterschleife kann man anhand einer rechteckigen Schleife, die sich parallel zu einem unendlich langen Leiter befindet, sehr anschaulich beschreiben. Wie in Anhang 1 von [Rod00] gezeigt, gibt es folgenden Zusammenhang zwischen M und der geometrischen Ausdehnung der rechteckigen Leiterschleife:

$$M = 0.2 \cdot c \cdot \ln \frac{b}{a} \ [\mu\text{H}].$$

Aus dieser Gleichung kann man ablesen, dass man M entweder durch größere räumliche Trennung oder durch kleinere räumliche Ausdehnung der Schleife verringern kann.

Weitere Gegenmaßnahmen sind das Verdrillen von Signalleitern, was zur Folge hat, dass sich das Vorzeichen des magnetischen Flusses ständig ändert und so der resultierende Fluss und die induzierte Spannung klein sind.

Von außen einwirkende magnetische Felder kann man auf zwei Arten mit einem Gehäuse abschirmen: zum einen mittels magnetischem Nebenschluss und zum anderen mit magnetischen Gegenfeldern. Die Abschirmung mittels eines hoch permeablen Werkstoffes nutzt das Prinzip des magnetischen Nebenschlusses aus. Dabei fließt ein Großteil des magnetischen Flusses ϕ durch die Gehäusewand, was zum Ergebnis hat, dass innerhalb des Gehäuses das magnetische Feld stark gedämpft ist. Der magnetische Nebenschluss funktioniert bei Gleichfeldern und bei geringen Frequenzen sehr gut, bei hohen Frequenzen nimmt die Schirmwirkung jedoch ab. Das hat zwei Gründe: zum einen ist der Skineffekt bei hoch permeablen Werkstoffen stark ausgeprägt, was die effektive Wandstärke verringert, zum anderen sind magnetische Werkstoffe schlechte elektrische Leiter, was bei höheren Frequenzen die Schirmwirkung verringert. Mit einem elektrisch gut leitenden Gehäuse erreicht man die Schirmwirkung durch Gegenfelder, die durch Wirbelströme in der Gehäusewand erzeugt werden. Unter einem Gegenfeld ist ein magnetisches Feld zu verstehen, welches dem störendem Feld entgegengerichtet ist und dieses dadurch kompensiert. Wirbelströme treten jedoch erst bei höheren Frequenzen auf und sind bei Gleichfeldern nicht vorhanden. Will man also Magnetfelder breitbandig abschirmen, muss man zwei geschachtelte Gehäuse verwenden, die jeweils gut elektrisch leitfähig und hoch permeabel sind. Für dieses Arbeit wurden elektrisch gut leitende Gehäuse verwendet. Auf die Schirmung von niederfrequenten Magnetfeldern wurde verzichtet, da man die Induktion durch geeignete schaltungstechnische Maßnahmen, wie kleine Leiterschleifen, einfacher reduzieren kann.

Kapazitive Kopplung

In einem Isolator zwischen zwei spannungsführenden Leitern baut sich stets ein elektrisches Feld auf. Handelt es sich um eine zeitlich veränderliche Spannung $u_1(t)$, so tritt im Isolator ein Verschiebungsstrom i_v auf:

$$i_v = C \cdot \frac{\mathrm{d}\, u_1}{\mathrm{d}\, t} \text{ bzw. } I_v = j\omega C \cdot U_1,$$

wobei C die Kapazität des parasitären Kondensators beschreibt.

Befinden sich nun zwei Leiter eines zweiten Schaltungsteils in diesem elektrischen Feld, so fließt ein Teil des Verschiebungsstroms i_v über diesen Schaltungsteil, was den Spannungsabfall $u_2(t)$ zur Folge hat. Der erste Schaltungsteil ist in diesem Fall die Störquelle, und der zweite Schaltungsteil ist die Störsenke. Abbildung 3.6 zeigt eine mögliche geometrische Anordnung (a) und Ersatzschaltbilder (b, c) der kapazitiven Kopplung.

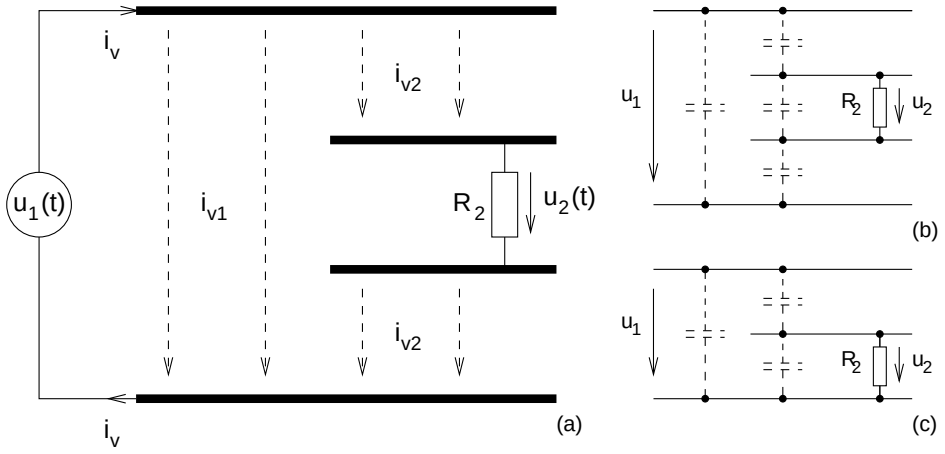

Abbildung 3.6: Kapazitive Kopplung, geometrische Anordnung (a), Ersatzschaltbild (b), Ersatzschaltbild mit gemeinsamen Leiter (c)

Die parasitären Kapazitäten, welche sich zwischen Störquelle und Störsenke befinden, werden als *Streukapazitäten* C_S bezeichnet. Durch Verkleinern dieser Streukapazitäten kann man die kapazitive Kopplung verringern. Eine Kapazität kann man prinzipiell durch Vergrößern des Abstandes zwischen den Kondensatorplatten bzw. durch das Verkleinern der Fläche der Kondensatorplatten erreichen. In unserem Fall heißt das, man trennt die entsprechenden Schaltungsteile räumlich, hält die gefährdeten Leitungen möglichst kurz und vermeidet parallel verlaufende Leitungen.

Eine weitere Gegenmaßnahme ist das Abschirmen der gefährdeten Schaltungsteile. Innerhalb der Abschirmung gibt es kein elektrisches Feld, welches einen Verschiebungsstrom zur Folge haben könnte. Der Störstrom fließt über das Gehäuse ab, welches möglichst gut elektrisch leitfähig sein soll. Abbildung 3.7 zeigt die Funktionsweise einer solchen Abschirmung. Der Verschiebungsstrom wird über den Schirm an der Störsenke vorbeigelenkt. Diese Form der Abschirmung wurde von Michael Faraday (1791–1867) entdeckt und wird deshalb auch *Faradayscher Käfig* genannt.

Ohmsche Kopplung

Eine ohmsche Kopplung tritt auf, wenn zwei Schaltungsteile (Störquelle und Störsenke) einen gemeinsamen Widerstand R_g nutzen. In Schaltplänen wird der Widerstand von Leitern als 0 Ω angenommen, in der Praxis haben Leiter jedoch einen Widerstand ungleich 0 Ω. Nutzen nun die Störquelle und die Störsenke einen Leiter gemeinsam, so erzeugt der Strom i_1 einen Spannungsabfall $u_g = R_g \cdot i_1$ am Widerstand R_g des gemeinsam genutzten Leiters (siehe Abbildung 3.8). Der Spannungsabfall kann die Störsenke beeinflussen. Die ohmsche

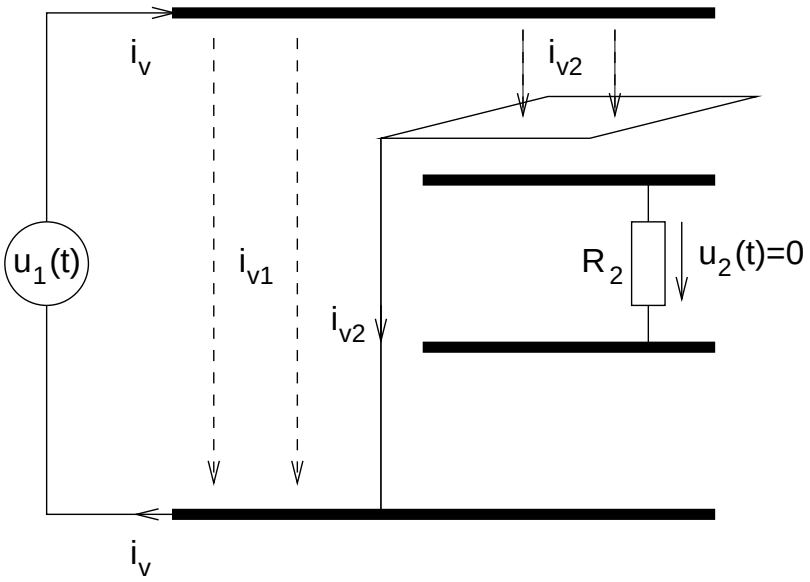

Abbildung 3.7: Abschirmung als Abhilfe gegen kapazitive Kopplungen

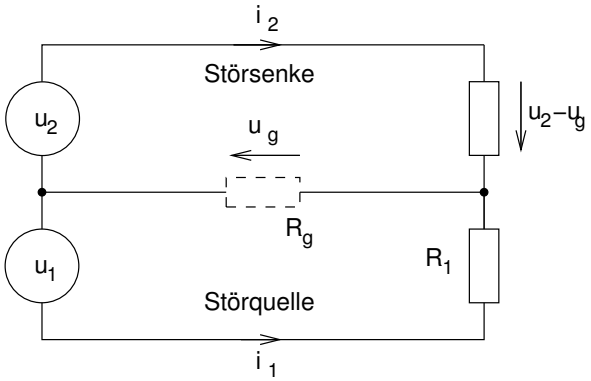

Abbildung 3.8: Ersatzschaltbild der ohmschen Kopplung

Kopplung tritt immer in Kombination mit einer induktiven Kopplung auf. Dabei entscheidet die Frequenz, welche Form der Kopplung dominierend ist. Bei niedrigen Frequenzen und Gleichstrom ist die ohmsche Kopplung dominierend, bei hohen Frequenzen ist die induktive Kopplung vorherrschend und die ohmsche Kopplung kann vernachlässigt werden.

Bei der induktiven Komponente handelt es sich um einen Sonderfall, bei dem Störquelle und Störsenke einen gemeinsamen Leiter nutzen. Wie bei der reinen induktiven Kopplung hängt die Gegeninduktivität von der Geometrie der Leiterschleife ab. Im Gegensatz zur reinen induktiven Kopplung kann man jedoch nicht den Abstand zwischen Störquelle und Störsenke verkleinern, lediglich die räumliche Ausdehnung der sekundären Schleife kann man verkleinern. Abbildung 3.9 veranschaulicht diesen Umstand.

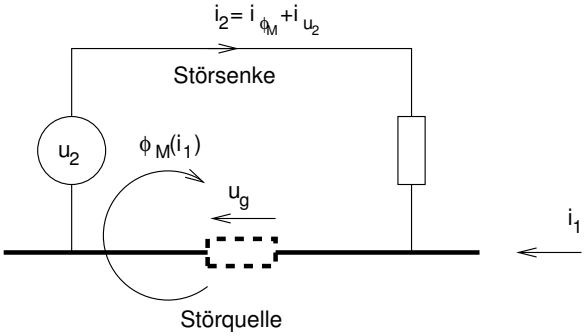

Abbildung 3.9: Ohmsche und induktive Kopplung treten immer gemeinsam auf

Ohmsch-induktive Kopplungen treten meist in Masseleitungen auf, da verschiedene Schaltungsteile eine gemeinsame Masseleitung als Rückleiter verwenden. Zur Vermeidung der ohmschen Kopplung muss man vor allem die gemeinsame Nutzung von Leitern vermeiden, was in sternförmigen Leiterstrukturen resultiert. Gegen die induktive Komponente helfen möglichst kurze Leiter und die räumliche Trennung von kritischen Schaltungsteilen.

Strahlungskopplung

Leitungen und Schlitze in räumlich ausgedehnten leitenden Strukturen, wie z. B. Gehäusen, können wie Antennen wirken und elektromagnetische Wellen aussenden und empfangen. Auf diese Art können Rundfunksender, aber auch Mobiltelefone und PCs Störspannungen bzw. Ströme in einer Schaltung hervorrufen. Strahlungskopplung kann durch Abschirmung, wie sie bei elektrischer, aber auch magnetischer Kopplung vorgestellt wurde, verhindert werden.

Kurzschlussprüfung

Ein sehr einfacher Test, um Fehler im Messaufbau zu entdecken, ist die *Kurzschlussprüfung*. Mit der Kurzschlussprüfung kann man feststellen, ob am Weg von der Messstelle zum Messgerät ein Fehler auftritt. Dabei wird einfach an der Stelle, an der der Tastkopf das Messsignal abgreift, der Tastkopf kurzgeschlossen. Die gemessene Spannung müsste dann 0 V sein. Ist die Spannung nicht 0 V, so muss auf dem Weg von der Messstelle zum Messgerät ein Fehler sein. Ein typischer Fehler kann eine Leiterschleife an der Messstelle sein, in die eine Spannung über induktive Kopplung induziert wird.

Zusammenfassung

Wie in den vorhergehenden Abschnitten gezeigt wurde, gibt es verschiedene Koppelmechanismen, denen man auf verschiedene Art und Weise entgegenwirken kann. Ein elektrisch gut leitendes Gehäuse kann elektrische, hochfrequente magnetische und elektromagnetische Felder aus der Umgebung gut abschirmen. Niederfrequente magnetische Felder kann man nur mit speziellen Gehäusen abschirmen, jedoch besteht die Möglichkeit, mit kleinen Leiterschleifen die Gegeninduktivität zu senken, was Kopplungen innerhalb der Schaltung ebenfalls verringert. Zur Vermeidung von Masseschleifen und daraus resultierenden ohmschen Kopplungen sollte man die Leitungen vom Messobjekt zu anderen Komponenten des Messaufbaus nach Möglichkeit galvanisch trennen. Innerhalb der Schaltung sind sternförmige Leiterstrukturen und große Abstände zwischen kritischen Schaltungsteilen zu empfehlen.

Im praktischen Teil der Arbeit wurde darauf geachtet, dass die Kopplungen zwischen verschieden Schaltungsteilen möglichst klein gehalten werden. Gegen Störungen aus der Umgebung wurden Datenleitungen galvanisch entkoppelt und die Messobjekte in Abschirmgehäuse eingebaut. Konkrete Messergebnisse, die die Leistungsfähigkeit der Maßnahmen zeigen, kann in den Kapiteln 4 bis 6 gefunden werden.

3.2.2. Stromaufnahme – Amplitude

Die Stromaufnahme von CMOS-Schaltungen ist mit den Daten korreliert, welche in der Schaltung verarbeitet werden. Es soll gezeigt werden, welcher Zusammenhang zwischen Daten und Stromaufnahme besteht und wie der Strom mit Hilfe eines Oszilloskops gemessen werden kann. Des weiteren soll gezeigt werden, welchen Einfluss die Digitalisierung auf das gemessene Signal nimmt. Die aufgestellten Thesen sollen anhand von Experimenten verifiziert werden.

Ersatzschaltung

In diesem Abschnitt soll eine CMOS-Ersatzschaltung und die Ersatzschaltung des Messaufbaus vorgestellt werden. Anhand dieses Modells soll gezeigt werden, wie Zwischenergebnisse der Berechnung mit der Stromaufnahme einer CMOS-Schaltung korrelieren.

Eine CMOS-Schaltung besteht aus einem Pullup- und einem Pulldown-Netzwerk. Beide Netzwerke sind zueinander komplementär, das bedeutet, dass immer wenn das Pullup-Netzwerk leitet, das Pulldown-Netzwerk sperrt und umgekehrt. Die einfachste CMOS-Schaltung ist der Inverter. Er besteht aus einem PMOS-Transistor im Pullup-Netzwerk und aus einem NMOS-Transistor im Pulldown-Netzwerk. Im stabilen Zustand fließt nur ein sehr kleiner Leckstrom von V_{dd} zu V_{ss}. Ein signifikanter Stromfluss entsteht bei CMOS-Schaltungen nur beim Umschalten von einem Zustand zum anderen. Der Umschaltstrom besteht aus zwei Komponenten. Die erste Komponente wird dadurch verursacht, dass beim Umschalten beide Transistoren kurz leitfähig sind und so quasi ein kurzzeitiger Kurzschluss verursacht wird. Die zweite und weitaus größere Komponente wird durch das Umladen der parasitären Kapazitäten der Verbindungsleitungen zwischen den Gattern verursacht.

Abbildung 3.10 zeigt eine CMOS-Schaltung mit einem hervorgehobenen Inverter inklusive parasitärer Lastkapazität, wie in [AO00] vorgestellt. In Tabelle 3.4 sind alle möglichen Bitübergänge aufgelistet und angegeben, welche Stromkomponenten bei den jeweiligen Bitübergängen fließen. Eine reale Schaltung, wie z. B. die eines Mikrocontrollers, besteht aus einer Vielzahl von solchen CMOS-Grundschaltungen, die alle ein ähnliches Verhalten wie der eben

vorgestellte Inverter an den Tag legen. Der Strom, den man von außen z. B. an einem Mikrocontroller messen kann, ist die Summe aller Ströme, die von den einzelnen CMOS-Gattern verursacht werden. Und doch ist es mit Hilfe der in Kapitel 2.3.3 vorgestellten Methoden möglich, den Zustand eines einzelnen Bits zu einem Zeitpunkt zu bestimmen.

Die Strommessung erfolgt indirekt über einen Spannungsabfall an einem Shunt-Widerstand R_m, der sich im Versorgungsstromkreis befindet. Über das ohmsche Gesetz berechnet sich der Strom aus dem Spannungsabfall über $i(t) = \frac{u(t)}{R}$. Die Spannung wird mit einem aktiven differentiellen Tastkopf abgegriffen, was Messungen von sehr hoher Genauigkeit ermöglicht.

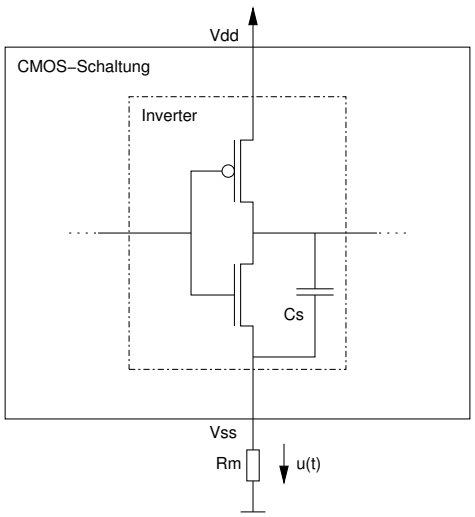

Abbildung 3.10: Ersatzschaltung des Messaufbaus zur Strommessung an einer CMOS-Schaltung

Tabelle 3.4: Stromfluss durch einen CMOS-Inverter

Übergang	Stromfluss
$0 \rightarrow 0$	nur Leckstrom
$0 \rightarrow 1$	Leckstrom, Kurzschlussstrom, Ladestrom
$1 \rightarrow 0$	Leckstrom, Kurzschlussstrom
$1 \rightarrow 1$	nur Leckstrom

Experimente zur Stromaufnahme

In diesem Abschnitt soll das Stromaufnahmemodell einer CMOS-Schaltung experimentell verifiziert werden. Als Messobjekt dient der ATmega32 Mikrocontroller, der in Kapitel 4 vorgestellt wird. Die Taktfrequenz des Mikrocontrollers beträgt 4 MHz, d. h. eine Taktperiode dauert $1/4 \cdot 10^{-6}$ s = 250 ns. Das Assembler-Programmfragment in Listing 3.1 zeigt den Teil des Programms, dessen Stromaufnahme mit dem Oszilloskop aufgezeichnet wurde und in Abbildung 3.11 dargestellt wird. Das Programm simuliert Bitübergänge, wie sie während der Abarbeitung von Algorithmen auftreten. Untersucht wird das Register `r24`, welches in Zeile 8

Listing 3.1: Programm zum Erzeugen des Bitübergangs (0x00→0xff)

```
1  sbi 50-0x20,7      ; Trigger-Pin setzen (2 Takte)
2
3  nop                ; 4 mal keine Operation
4  nop                ; (4 Takte)
5  nop
6  nop
7
8  ldi r24, 0         ; Register mit 0x00 laden (1 Takt)
9  ldi r24, 255       ; Register mit 0xff laden (1 Takt)
10
11 nop                ; 4 mal keine Operation
12 nop                ; (4 Takte)
13 nop
14 nop
15
16 cbi 50-0x20,7      ; Trigger-Pin loeschen (2 Takte)
```

auf einen Startwert und in Zeile 9 auf einen Zielwert gesetzt wird, der die Bitübergänge hervorruft. Für das Experiment wurden im Register jeweils 8 Bitübergänge zugleich hervorgerufen, d. h. es wurden alle Bits im Register gekippt, damit die maximale Stromaufnahme erzeugt wird. Im Programmfragment ist der Übergang (0x00→0xff) exemplarisch dargestellt, das Experiment wurde jedoch für alle vier möglichen Bitübergänge durchgeführt: (0x00→0xff), (0xff→0xff), (0x00→0x00) und (0xff→0x00). Um das Signal zu glätten, wurden für jeden der vier Übergänge 100 Messungen durchgeführt, woraus letztendlich Mittelwerte gebildet wurden. Das Ergebnis der Mittelwertbildung sieht man in Abbildung 3.11 oben. Der gezeigte Zeitausschnitt entspricht genau der Zeit zwischen den zwei Triggerflanken, die von den Befehlen `sbi` und `cbi` erzeugt werden. Die Befehle `sbi` und `cbi` dauern zwei Taktperioden, wobei das Triggersignal in der zweiten Hälfte der Abarbeitung des Befehls gesetzt bzw. gelöscht wird. Im Diagramm in Abbildung 3.11 sind deshalb die beiden Befehle abgeschnitten.

Deutlich sind die vier `nop`-Befehle vor und nach dem Kopierbefehl zu sehen. Auffallend ist dabei, dass `nop`-Befehle, die an andere Befehle grenzen, auch eine größere Stromamplitude aufweisen. Das ist vermutlich auf das Pipelining und auf verschiedene Zustände der Busse im Mikrocontroller zurückzuführen. Die Datenabhängigkeit beim `ldi`-Befehl ist eindeutig zu sehen, wenn man den vergrößerten Ausschnitt der Stromkurve in Abbildung 3.11 unten betrachtet. In diesem Bild sieht man auch, dass die Stromaufnahme bei einem Bit-Übergang (0x00→0xff) am größten ist. Dieses Ergebnis wird auch vom CMOS-Modell im vorhergehenden Abschnitt vorhergesagt. Auffallend ist jedoch der Bitübergang (0x00→0x00), welcher gemäß CMOS-Modell keine erhöhte Stromaufnahme haben sollte. Der Bit-Übergang (0xff→0x00) hat gegenüber dem Modell eine zu kleine Stromaufnahme. Um Messfehler auszuschließen, wurde das Experiment mehrmals mit verschiedenen Varianten des Programms wiederholt. Dabei wurde das Setzen des Ausgangszustands des Registers `r24` an verschiedenen Stellen vorgenommen und in dem oben angegebenen Programm wurde eine `nop` Operation zwischen dem Setzen des Ausgangszustands von `r24` und dem Hervorrufen des Bitübergangs eingefügt. Alle Messungen hatten das gleiche Ergebnis zur Folge, womit eine Abhängigkeit

vom Programmcode weitgehend ausgeschlossen wurde. Als möglichen Grund für diese Abweichung kann man vermuten, dass das Modell, welches sich auf ein einziges CMOS-Gatter bezieht, verschiedene parallel ablaufende Schaltvorgänge, wie sie in einer komplexen Schaltung vorkommen, nicht berücksichtigt.

Für DPA-Analysen ist das Modell geeignet, weil man die datenabhängige Stromaufnahme der CMOS-Schaltung sehr gut vorhersagen kann. Voraussetzung dafür ist, dass der ursprüngliche Zustand eines Bits logisch Null war. Wenn der ursprüngliche Zustand logisch Eins war, ist das Modell nicht zuverlässig, was durch die Mittelwertbildung bei der DPA-Analyse kompensiert wird.

Die Abweichung, welche durch den unbekannten ursprünglichen Zustand des Bits hervorgerufen wird, kann als Rauschen aufgefasst werden. In der Einleitung zu Abschnitt 3.2 wurden mögliche Rauschquellen in fünf Kategorien eingeteilt, wobei diese Form des Rauschens zum algorithmischen Rauschen gezählt wird.

Da jetzt experimentell bewiesen wurde, dass der Übergang von einem Null-Bit auf ein Eins-Bit den größten Stromfluss verursacht, soll nun in einem zweiten Experiment gezeigt werden, wie sehr die Stromaufnahme vom Hamminggewicht abhängt, wenn der Startwert des Bytes 0x00 ist. Wir haben ein ähnliches Programm wie im vorigen Experiment vorliegen, mit dem Unterschied, dass in Zeile 9 der Zahlenwert, welcher in das Register kopiert wird, nicht konstant ist. In diesem Programm (siehe Listing 3.2) werden die Bitübergänge im Register `r30` untersucht. Im Register `r24` steht ein Zahlenwert, der über die serielle Schnittstelle empfangen wurde und der in Zeile 9 in das zu untersuchende Register `r30` kopiert wird.

Listing 3.2: Programm für die Analyse der Stromaufnahme bei verschiedenen Hamminggewichten

```
1  sbi 50-0x20,7      ; Trigger-Pin setzen (2 Takte)
2
3  nop                ; 4 mal keine Operation
4  nop                ; (4 Takte)
5  nop
6  nop
7
8  ldi r30, 0         ; Register loeschen (1 Takt)
9  mov r30, r24       ; Empfangenden Wert in Register schreiben
10                    ; (1 Takt)
11 nop                ; 4 mal keine Operation
12 nop                ; (4 Takte)
13 nop
14 nop
15
16 cbi 50-0x20,7      ; Trigger-Pin loeschen (2 Takte)
```

Die über die serielle Schnittstelle gesendeten Bytes werden in Matlab mittels eingebautem Zufallszahlengenerator erzeugt und sind gleichverteilt. Da man auf diese Art nur sehr wenige Zahlen mit Hamminggewicht von 0 oder 8 bekommt, wurden zusätzlich die Zahlen 0 und 255 hundert Mal gesendet, so dass für jedes Hamminggewicht 100–400 Messkurven vorlagen, aus denen die Mittelwerte gebildet wurden. Abbildung 3.12 zeigt die Gegenüberstellung der Messungen zum Zeitpunkt des Befehls `mov r30,r24` bei verschiedenen Hamminggewichten

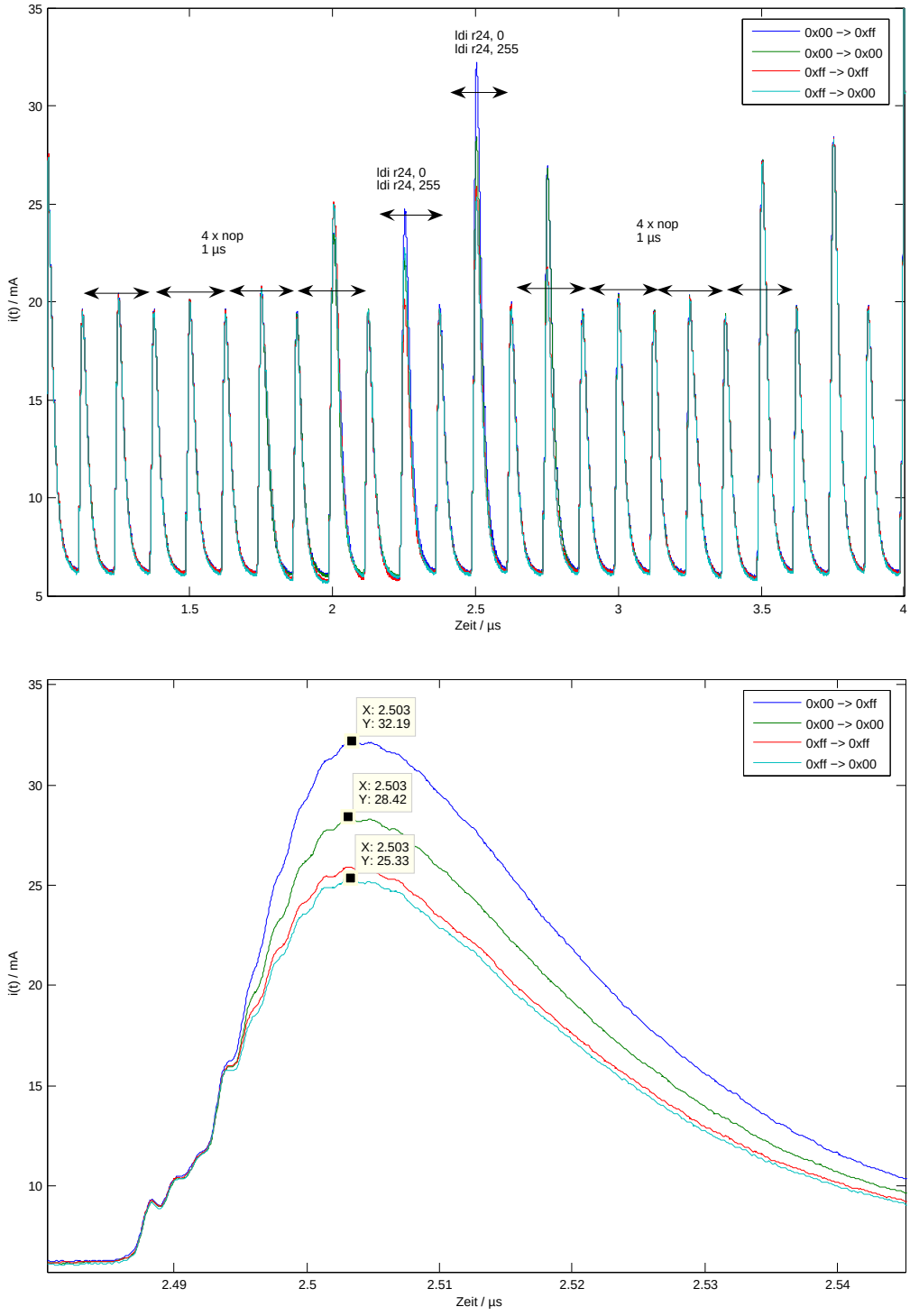

Abbildung 3.11: Stromaufnahme bei Abarbeitung des Testprogramms aus Listing 3.1 mit verschiedenen Bitübergängen. Die obere Kurve zeigt die Stromaufnahme zwischen den beiden Triggerpunkten. Die untere zeigt die Stromaufnahme des ldi-Befehls alleine.

des empfangenen Bytes. Es ist sehr deutlich zu sehen, dass bei steigendem Hamminggewicht die Stromaufnahme proportional steigt.

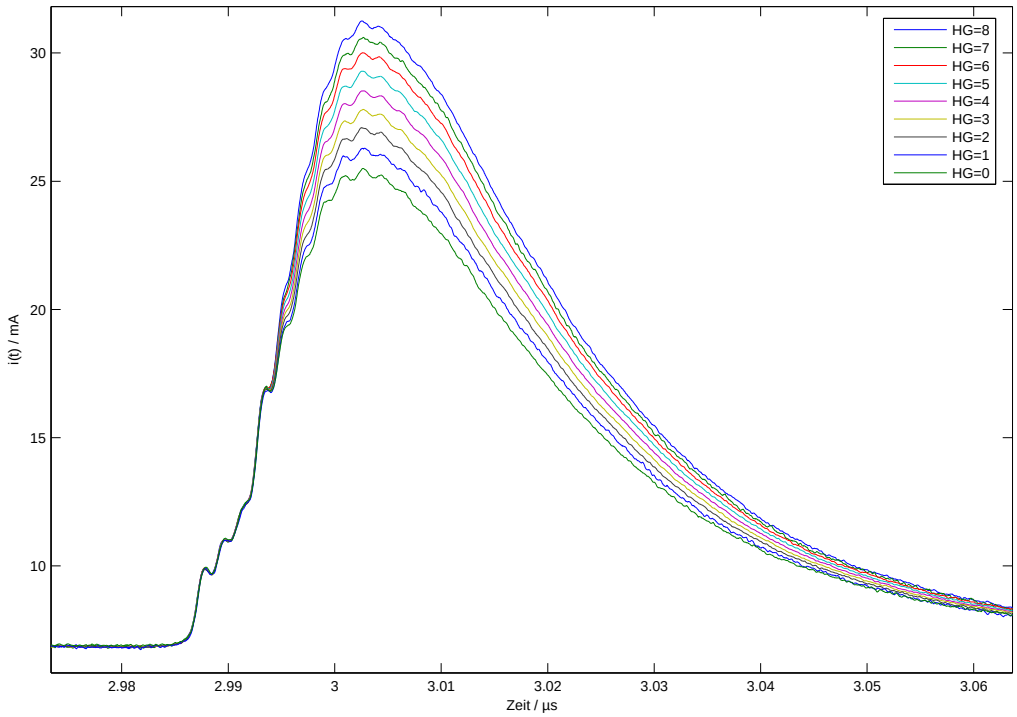

Abbildung 3.12: Gegenüberstellung der Stromaufnahme bei Bitübergängen ($0 \rightarrow 1$) und unterschiedlichen Hamminggewichten

Quantisierungsrauschen

Die Differenz eines wertkontinuierlichen Signals und seines zugehörigen wertdiskreten Signals wird als Quantisierungsrauschen bezeichnet. Es kann in die zwei Teile *granulares Rauschen* und *Überlastungsrauschen* gegliedert werden. Hier soll nur das granulare Rauschen betrachtet werden, das bei gleichmäßiger Quantisierung nur von der Stufenhöhe Δ abhängt. Die Stufenhöhe bezeichnet dabei den Abstand zwischen zwei Quantisierungsstufen. Das Überlastungsrauschen, welches bei Übersteuerung des Quantisierungsbereichs auftritt, soll unbeachtet bleiben, da der Spannungsbereich am Oszilloskop so eingestellt wird, dass es nicht zu Übersteuerungen kommt. Abbildung 3.13 gibt einen Überblick über diese Zusammenhänge.

Der quadratische Mittelwert der Rauschspannung $\overline{q^2}$ ist gleich der Varianz der Rauschspannung σ_Q^2 und gleich der Rauschleistung P_Q

$$P_Q = \sigma_Q^2 = \overline{q^2} = \int_{-\Delta/2}^{\Delta/2} q^2 \cdot \frac{1}{\Delta}\, dq = \frac{\Delta^2}{12}.$$

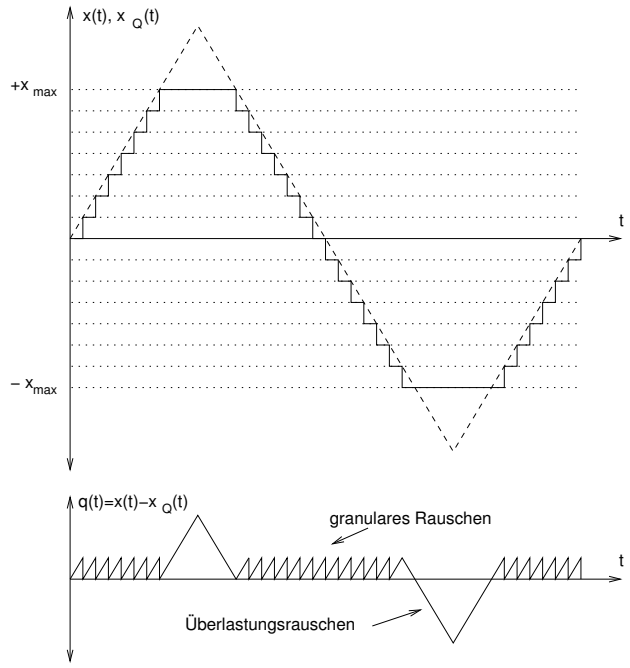

Abbildung 3.13: Gegenüberstellung des wertkontinuierlichen und des wertdiskreten Signals sowie der Differenz der beiden, die mit Quantisierungsrauschen bezeichnet wird.

Die mittlere Signalleistung P_x bei einem gleichverteilten Signal $x(t)$ beträgt

$$P_x = \overline{x^2} = \mathrm{E}\{x^2\} = \int_{-x_{max}}^{x_{max}} x^2 \cdot \frac{1}{2x_{max}}\, dx = \frac{x_{max}^2}{3}.$$

Der Signal-zu-Quantisierungsrauschabstand (Signal to Noise Ratio, SNR) ist das Verhältnis der Signalleistung zur Rauschleistung

$$\mathrm{SNR}_Q = \frac{P_x}{P_Q} = \frac{x_{max}^2/3}{\Delta^2/12} = 2^{2m},$$

wobei m die Bittiefe des Quantisierers ist. In der Regel wird der SNR in Dezibel angegeben

$$\mathrm{SNR}_Q^{\mathrm{dB}} = 10 \cdot \log(\mathrm{SNR}_Q)\mathrm{dB} = 20 \cdot m \cdot \log(2)\mathrm{dB} = m \cdot 6\mathrm{dB}.$$

Man kann also zusammenfassend sagen, dass bei einem gleichverteilten Signal der Signal-zu-Quantisierungsrauschabstand sich mit jedem zusätzlichen Bit des Quantisierers um 6 dB erhöht. Im Fall eines acht Bit Quantisierers, wie er beim digitalen Speicheroszilloskop zum Einsatz kommt, hat man ein $\mathrm{SNR}_Q^{\mathrm{dB}} = 6 \cdot 8$ dB $= 48$ dB. Das entspricht einem SNR_Q von 65536. Das Quantisierungsrauschen hat also einen sehr geringen Einfluss auf das Ergebnis einer DPA-Analyse.

Abschätzung des gesamten SNR bei DPA-Messungen

Eine allgemein gültige Aussage über den SNR ist schwer zu treffen, da sehr viele Parameter den SNR beeinflussen. In der Dissertation von Messerges [Mes00a] wurde eine Näherung für

den SNR vorgeschlagen, die jedoch auf so vielen Annahmen basiert, dass sie schwer anwendbar ist.

Es ist schwer, einen Zusammenhang zwischen Signalqualität und Anzahl der benötigten Messungen zu finden. Denn die verschlüsselten Daten, die angegriffene Struktur innerhalb des Algorithmus und der geheime Schlüssel haben einen sehr großen Einfluss auf die benötigte Anzahl der Messungen. Der Unterschied bei ein und demselben Messdatendatensatz kann mehr als Faktor 10 betragen. Eine genaue Betrachtung dieses Phänomens würde den Rahmen dieser Diplomarbeit sprengen.

3.2.3. Überlegungen zur Zeitachse – Ausrichten der Stromkurven

Bei DPA-Angriffen werden viele Messungen hintereinander durchgeführt. Aus den einzelnen Messreihen werden, nach der Einteilung in verschiedene Mengen, Mittelwerte gebildet. Bei dieser Mittelwertbildung ist es sehr wichtig, dass die Messreihen die gleiche zeitliche Ausrichtung haben. Ist das nicht gegeben, werden die Messergebnisse durch die Mittelwertbildung verfälscht, ohne dass das Rauschen reduziert wird. Das Erschweren bzw. Verhindern der exakten zeitlichen Ausrichtung wird als Gegenmaßnahme eingesetzt (z. B. Random Wait States).

In Laboraufbauten kann man das zu evaluierende Programm dahingehend modifizieren, dass zu einem definierten Zeitpunkt ein externes Triggersignal gesetzt wird. Das ist jedoch nur bei Mikrocontrollern mit einem freien I/O-Pin möglich. Bei Chipkarten fällt diese Möglichkeit auch bei Evaluierungen eines Algorithmus weg, da keine freien I/O-Pins verfügbar sind. Man kann zwar mit Hilfe der Datenübertragung zwischen Chipkarte und Terminal einen Startzeitpunkt für die Messung festlegen, jedoch ist der Startzeitpunkt zu ungenau. Das liegt daran, dass es zwar möglich ist festzustellen, wann ein Befehl bei der Chipkarte angekommen ist. Wann die Berechnung der kryptographischen Operation beginnt, kann man aber nicht exakt feststellen. Deshalb muss man einen anderen Ansatz wählen, um eine zeitliche Ausrichtung der Messkurven zu erreichen. Eine nahe liegende Methode ist das Suchen einer bestimmten Sequenz, die in allen Stromkurven zur gleichen Zeit vorkommt. Anhand dieses Musters kann man die zeitlichen Verschiebungen zwischen den Kurven bestimmen und ausgleichen.

Hat der Algorithmus „Random Wait States“ als Gegenmaßnahme eingebaut, so wird die Aufgabe der zeitlichen Ausrichtung ungleich schwieriger. Bei dieser Gegenmaßnahme werden an verschiedenen Stellen des Programms zufällige Wartezeiten eingefügt. Beim zeitlichen Ausrichten muss man also in der Stromkurve Abschnitte finden, die zur gleichen Codesequenz gehören, damit man sie exakt übereinander legen kann.

Pattern-Matching

In diesem Abschnitt soll gezeigt werden, wie mit der Kreuzkorrelationsfunktion die zeitliche Verschiebung zwischen Messkurven ausgeglichen werden kann. Seien $x = \{x_n\}, n = 1, \ldots, N$ und $y = \{y_n\}, n = 1, \ldots, N$ zwei Zahlenfolgen. Die nicht normierte Kreuzkorrelationsfunktion definiert durch

$$R_{xy}(m) = \begin{cases} \sum_{n=0}^{N-m-1} x_n \cdot y_{n+m} = \sum_{n=0}^{N-m-1} x_{n+m} \cdot y_n & \text{für } m \geq 0, \\ R_{xy}(-m) & \text{für } m < 0 \end{cases}$$

ist ein Maß für die Abhängigkeit der zwei Zahlenfolgen x und y. Die Größe m bezeichnet dabei die Verschiebung der beiden Zahlenfolgen gegeneinander. Bei der Verschiebung m, für die $R_{xy}(m)$ maximal wird, besteht die größte Abhängigkeit zwischen den beiden Zahlenfolgen. Man kann also die Kreuzkorrelationsfunktion für das Suchen einer bestimmten Sequenz in einer Messkurve verwenden. Ist $x = y$, so spricht man von der Autokorrelationsfunktion, sie hat das Maximum an der Stelle $m = 0$.

Gegeben ist ein Satz von N Messreihen mit M Messpunkten $I_{ij}, i = 1, \ldots, N, j = 1, \ldots, M$. Die Grundidee besteht darin, eine Sequenz in der ersten Messreihe auszuwählen, die in allen Messreihen vorhanden ist. Diese Sequenz wird mittels Kreuzkorrelationsfunktion in jeder weiteren Messreihe gesucht. Die Stromaufnahme ist wegen des Prozessortaktes nahezu periodisch. Dadurch kann es passieren, dass vom Korrelationskoeffizienten zwar eine hohe Abhängigkeit angezeigt wird, die gefundene Sequenz in der Stromaufnahme jedoch nicht zur gleichen Programmsequenz gehört. Deshalb sollte die gewählte Sequenz charakteristische Muster, wie z. B. ausgeprägte Stromspitzen, aufweisen. Die Länge der Suchsequenz wurde jeweils so gewählt, dass mindestens 20–200 Prozessortakte abgedeckt werden. Die Erfahrung hat gezeigt, dass bei unzuverlässiger Suche, das Vergrößern des Fensters eine höhere Zuverlässigkeit zur Folge hatte, ein zu großes Fenster jedoch die Rechenzeit vergrößert und die Zuverlässigkeit wieder sinken lässt.

In der Praxis will man erreichen, dass die Messreihen nach dem Verschieben noch immer die gleiche Länge aufweisen. Das kann man nur dadurch erreichen, dass man alle Kurven, nachdem man die zeitliche Verschiebung berechnet hat, auf die gleiche Länge zuschneidet. Des Weiteren darf man die Sequenz, die man sucht, nicht am Anfang einer Messreihe wählen, da sonst bei negativer Verschiebung einer weiteren Messreihe die Sequenz nicht mehr gefunden werden kann. Die Lösung des Problems ist das Definieren eines Fensters, in dessen Mitte die Suchsequenz gelegt wird. Auf diese Art kann man zusätzlich die Rechenzeit verringern, da man nicht die gesamte Messreihe durchsuchen muss. Abbildung 3.14 zeigt den Zusammenhang zwischen Rohdaten, Suchfenster, Suchsequenz und den resultierenden zeitlich ausgerichteten und gleich langen Messreihen. Die Wahl der Fenstergröße `Flength` hängt von der zu erwartenden Verschiebung zwischen den einzelnen Messkurven ab. Es sollte jedoch möglichst klein sein, da ein zu großes Fenster auch die resultierenden Messreihen mehr verkürzt.

Aus den oben genannten Gründen ist folgende Vorgehensweise zu empfehlen:

- Schon beim Messen sollte man sich optisch eine charakteristische Sequenz auf dem Display des Speicheroszilloskops aussuchen. Nach Möglichkeit sollte diese Sequenz in der Mitte eines gewählten Suchfensters am Anfang der Messkurve liegen. Es werden N Messkurven mit dem Speicheroszilloskop gemessen und zum Laborrechner übertragen.

- In der ersten Messkurve wird die Suchsequenz mit der Länge von 20–200 Prozessortakten ausgewählt.

- In weiteren Kurven wird das Muster mittels Kreuzkorrelationsfunktion gesucht. Die entsprechenden Kurven werden gegeneinander verschoben und die überschüssigen Messpunkte werden, wie in Abbildung 3.14, verworfen.

- Mehrere gegeneinander verschobene Kurven können nun mittels Plot-Befehl in einem Diagramm ausgegeben werden. Durch die genaue Betrachtung (Zoom) von charakteristischen Stellen kann man deutlich erkennen, ob die Verschiebung richtig erfolgt ist.

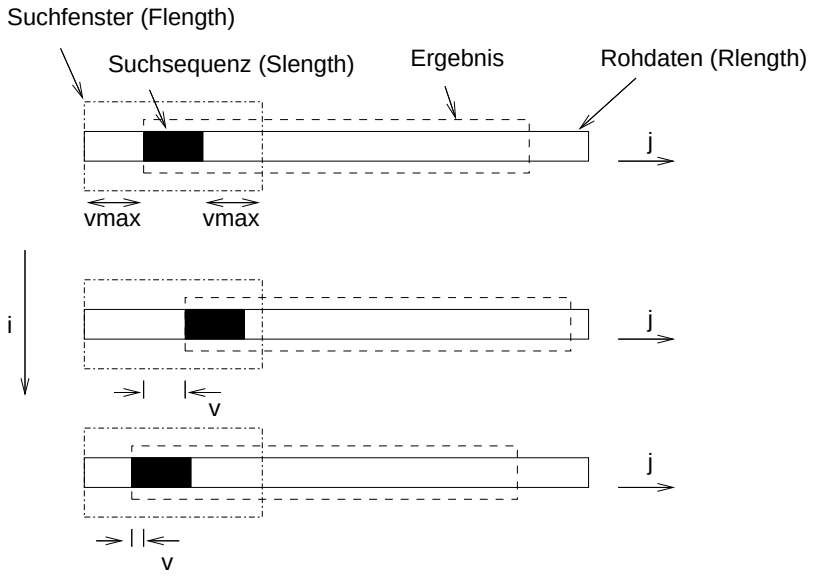

Abbildung 3.14: Zusammenhang zwischen Rohdaten, Suchfenstern, Suchsequenzen und dem Ergebnis

- Wenn die Verschiebung *nicht* erfolgreich war, verlängert man die Suchsequenz und wiederholt die Suche in den weiteren Messkurven.

- Wenn die Verschiebung erfolgreich war, fährt man mit der weiteren Verarbeitung der Messdaten fort bzw. man führt die DPA-Analyse durch.

Listing 3.3 zeigt in Form von Pseudocode die Vorgehensweise bei der Verwendung von Matlab. Die Bezeichnungen stimmen mit denen aus Abbildung 3.14 überein. In Zeile 1 wird die Suchsequenz aus dem ersten Rohdatensatz gewonnen. Dabei wird der „(von:bis)“ Operator verwendet. Dieser Operator liefert den Ausschnitt eines Vektors zurück, der von den Indizes „von“ und „bis“ begrenzt wird. In Zeile 3 wird die Kreuzkorrelation berechnet und das Maximum in den berechneten Korrelationskoeffizienten gesucht. Die Berechnung der Kreuzkorrelation erfolgt von Matlab im Frequenzbereich und benötigt wenig Rechenzeit. Die Berechnung im Frequenzbereich ist deshalb so schnell, weil alle Korrelationskoeffizienten auf einmal berechnet werden und nicht sukzessive für jede Verschiebung m berechnet werden müssen. Damit beide Vektoren die gleiche Länge wie das Suchfenster aufweisen, wird der kürzere Vektor von der `xcorr`-Funktion automatisch mit Nullen aufgefüllt. Die `max`-Funktion sucht das Maximum im Ergebnis der `xcorr`-Funktion und gibt als Resultat den Index (idx) und den Wert (val) des Maximums zurück. Das Ergebnis der xcorr Funktion entspricht jedoch nicht genau der mathematischen Definition der Kreuzkorrelation. Das liegt daran, dass es in Matlab keine negativen Indizes gibt, das Ergebnis Kreuzkorrelation aber sehr wohl für negative Indizes bzw. Verschiebungen m gültig ist. Das wird dadurch umgangen, dass das Ergebnis von `xcorr` so weit verschoben wird, bis keine negativen Indizes mehr auftreten. Diese Verschiebung wird in Zeile 4 wieder rückgängig gemacht, indem die Fensterlänge Flength um 1 verkleinert vom gefundenen Index des Maximums (idx) abgezogen wird. In Zeile 6 wird schlussendlich die Kurve mit der richtigen zeitlichen Ausrichtung und der richtigen Länge ermittelt.

Listing 3.3: Matlab-Pseudocode, für die Suche eines Musters in einer Messkurve

```
1 Suchsequenz = Rohdaten_1(vmax:Slength+vmax);
2
3 [val,idx]=max(xcorr(Rohdaten_i(1:Flength), Suchsequenz);
4 v = idx - (Flength-1);
5
6 Ergebnis_i = Rohdaten_i(vmax+v:v+Rlength-vmax)
```

Verbessertes Pattern-Matching

Instabilitäten des Oszillators des Messobjekts verursachen gelegentlich Probleme beim Übereinanderlegen der Messkurven. Das Problem besteht darin, dass zwei Messkurven zwar zu Beginn der Kurven richtig übereinander liegen, jedoch laufen sie mit der Zeit um z. B. eine halbe Taktperiode auseinander. Abbildung 3.15 veranschaulicht dieses Problem. In diesem Diagramm sind zwei verschiedene Ausschnitte zweier übereinandergelegter Stromkurven dargestellt. Im rechten Ausschnitt sieht man die Verschiebung zwischen den beiden Kurven. Diese Verschiebung hat verheerende Auswirkungen auf die Mittelwertbildung, da zeitlich nicht übereinstimmende Messwerte in die Mittelwertbildung einfließen. Im Rahmen der Diplomarbeit wurde eine Methode entwickelt, um ein solches Auseinanderlaufen zu erkennen. Bei dieser Methode wird das Pattern-Matching zu Beginn und am Ende der Messkurven vorgenommen. Anschließend wird das Ergebnis verglichen. Bei einer einstellbaren Abweichung wird die Messkurve verworfen und fließt nicht in die DPA-Analyse ein. Ein Strecken bzw. Stauchen von Messkurven wurde auf Grund von zu großem Rechenaufwand nicht umgesetzt.

Oversampling und synchrone Messung

Für die Messungen während dieser Diplomarbeit wurden die Oszillatoren des Messobjekts und des Oszilloskops nicht synchronisiert. Dabei stößt man auf das Problem einer sich ändernden Phasenverschiebung zwischen dem Takt des Messobjekts und dem Sampling-Takt des Oszilloskops (Phasenjitter). Abbildung 3.16 illustriert diese Problematik. In diesem Beispiel wurde das gleiche Signal dreimal hintereinander mit gleicher Abtastperiode T abgetastet. Bei der synchronen Messung haben alle drei Messungen die gleiche Messreihe zur Folge, somit ist der Mittelwert ebenfalls identisch. Bei der asynchronen Abtastung variiert der Abtastzeitpunkt, was in verschiedenen Messreihen resultiert. Das wirkt sich auf den Mittelwert aus, der durch die verschiedenen Abtastzeitpunkte verfälscht wird. Der Fehler, der dabei auftritt, kann als Rauschen aufgefasst werden und verschlechtert somit das Signal-Rausch-Verhältnis. Durch die Verkleinerung der Abtastperiode T, d. h. durch die Erhöhung der Abtastfrequenz des Oszilloskops (Oversampling), kann man den Fehler, der durch Phasenjitter verursacht wird, verkleinern, weil die Varianz der Phasenänderung kleiner wird. Mit 70–625 Messpunkten pro Prozessortaktperiode wurden bei den Messungen auf den verschiedenen Messobjekten gute Ergebnisse erzielt. Die vielen Messpunkte haben einen weiteren sehr wichtigen Vorteil zur Folge, nämlich dass das Pattern-Matching bessere Ergebnisse liefert. Die vielen Messpunkte werden für die eigentliche DPA-Analyse nicht benötigt, da die gesuchte Information hauptsächlich in der Grundfrequenz und in der ersten Oberwelle des Prozessortaktes zu finden ist. Deshalb werden die Daten, bevor sie einer DPA-Analyse unterzogen werden, gefiltert und einem Down-

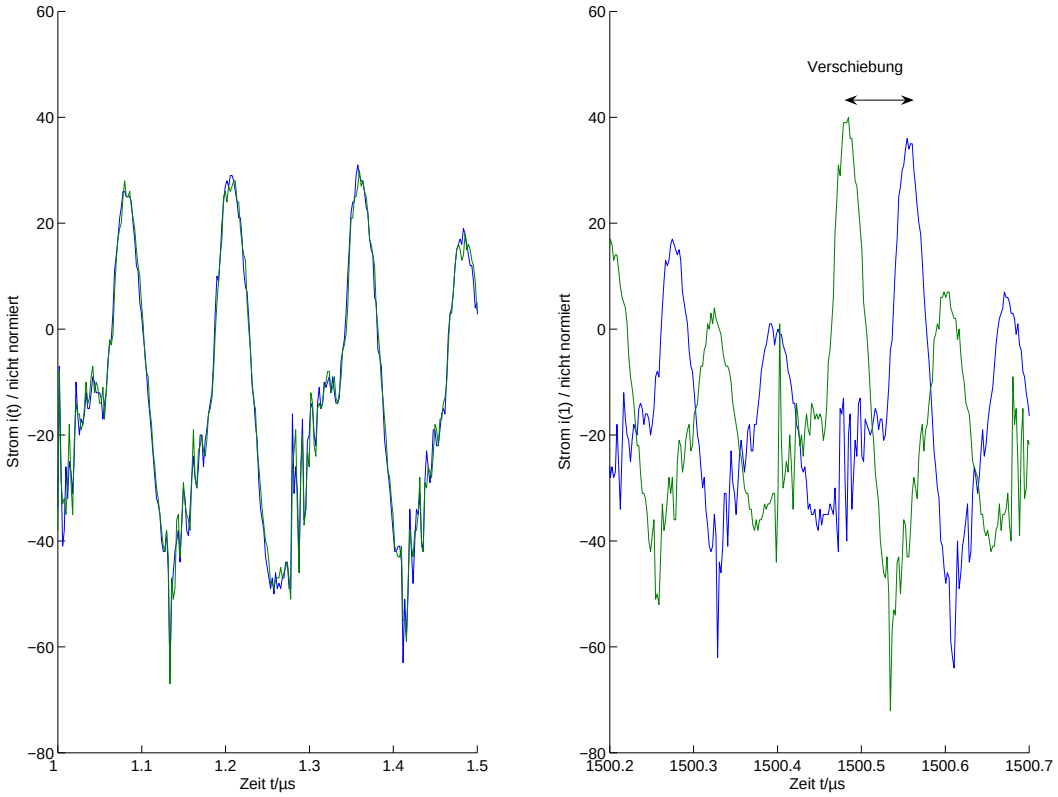

Abbildung 3.15: Auseinanderlaufen zweier Messkurven bei Instabilitäten des Oszillators

sampling unterzogen. Dies erfolgt mit dem Matlab-Befehl `decimate` aus der Signal Processing Toolbox. Dieser Befehl filtert die Messdaten mittels eines Tschebyscheff-Tiefpassfilters achter Ordnung und reduziert die Abtastfrequenz um einen einstellbaren Faktor. Dadurch wird höherfrequentes Rauschen unterdrückt und durch die verringerte Datenmenge die DPA-Analyse erheblich beschleunigt.

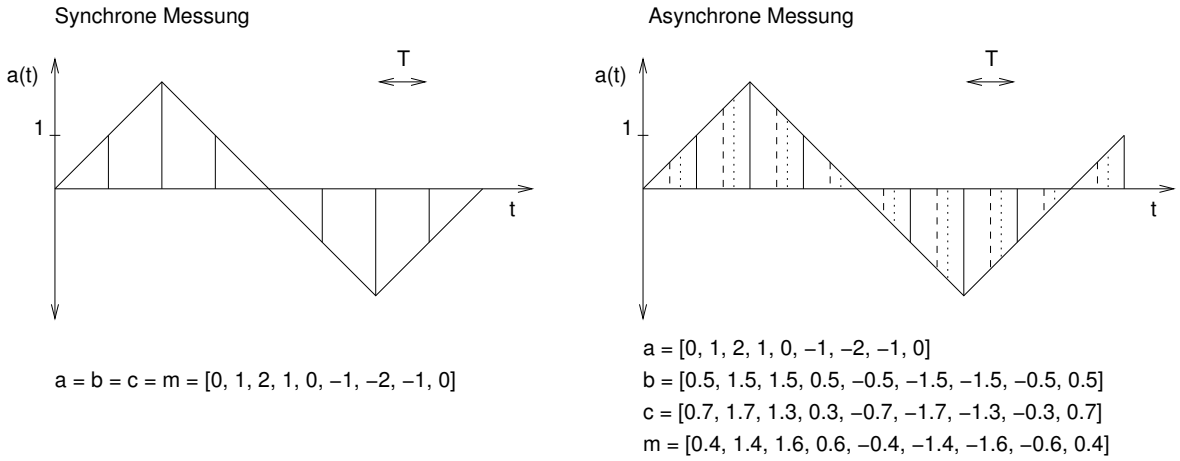

Abbildung 3.16: Gegenüberstellung von synchroner und asynchroner Messung

Eine synchrone Messung erfordert ein Oszilloskop mit einem Eingang für einen externen Sampling-Takt. Dieser externe Sampling-Takt wird von einem Oszillator erzeugt, der zugleich mit einem Frequenzteiler verbunden ist. Dieser Frequenzteiler erzeugt einen Takt mit einer um den Faktor n verringerten Frequenz. Dieser Takt dient als Prozessortakt des Messobjekts. Der Teilerfaktor n bestimmt die Anzahl der Messpunkte pro Prozessortakt. Messungen für diese Diplomarbeit haben gezeigt, dass bei Softwareimplementierungen auf Mikrocontrollern, die Seitenkanalinformation hauptsächlich in der zweiten Oberwelle des Prozessortaktes zu finden ist. Damit das Abtasttheorem eingehalten werden kann, muss also die Stromaufnahme mit einer mindestens viermal höheren Frequenz als der Prozessortakt abgetastet werden. D. h. der Teilerfaktor n muss einen Wert von mindestens 4 aufweisen. Abbildung 3.17 illustriert anhand eines Blockschaltbildes die Funktionsweise der synchronen Messung.

3.2.4. Verbesserungen der Signalqualität

Im Laufe dieser Arbeit wurden verschiedene Verbesserungen des Messsetups vorgenommen, um die Signalqualität der Stromkurven zu erhöhen. In diesem Abschnitt sollen die einzelnen Verbesserungen kurz aufgezeigt werden.

Lichtwellenleiter beim ATmega32-Board

Eine der ersten Ideen war es, die serielle Schnittstelle des zu untersuchenden Mikrocontrollers vom PC zu entkoppeln. Dies hat zwei positive Effekte zur Folge:

Die serielle RS232-Schnittstelle verwendet andere Pegel für die Übertragung der digitalen Daten als herkömmliche digitale Schaltungen. Aus diesem Grund benötigt man einen Pegelwandler, wenn man einen Mikrocontroller mit der seriellen Schnittstelle eines PCs verbinden

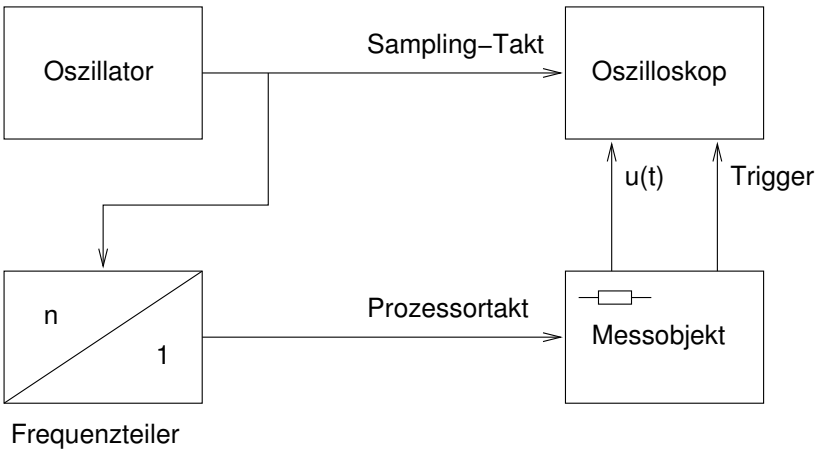

Abbildung 3.17: Messaufbau für eine synchrone DPA-Messung

will. Der verwendete Pegelwandler mit der Bezeichnung MAX232 hat beim ersten Testaufbau mit dem ATmega-Mikrocontroller Störungen verursacht. Der Grund für diese Störungen ist die schwingende Ladungspumpe, die die Spannungspegel vergrößert. Erst durch eine größere räumliche Trennung von Pegelwandler und Mikrocontroller, d. h. durch eine Verkleinerung der Kopplungen, konnte der Störeinfluss verringert werden. Mit der Einführung der Übertragung mittels Lichtwellenleiter (LWL) konnte der Pegelwandler auf der Seite des Mikrocontrollers komplett entfallen, was diese Form der Störung komplett ausgeschaltet hat und so das externe Rauschen stark verringert hat.

Der zweite positive Effekt ist die galvanische Entkopplung des zu untersuchenden Mikrocontrollers vom PC, was Masseschleifen und somit Kopplungen vom PC zum Mikrocontroller unterbindet. Den gleichen Effekt kann man auch mit einem Optokoppler erzielen, jedoch hat ein Optokoppler durch die geringen Abmessungen den Nachteil, dass kapazitive und induktive Kopplungen zwischen den galvanisch getrennten Teilen auftreten können.

Dieses Beispiel mit dem Pegelwandler zeigt, dass man in der zu evaluierenden Schaltung nach Möglichkeit alle nicht benötigten aktiven Komponenten vermeiden soll. Durch eine räumliche Trennung des Mikrocontrollers und nicht vermeidbarer Komponenten kann man Kopplungen verkleinern.

Abschirmung mit Gehäuse

Ein Gehäuse kann je nach Ausführung elektrische, magnetische und elektromagnetische Felder abschirmen. Für unseren konkreten Fall wurde eine Abschirmung mit elektrisch gut leitendem Material (Aluminium) gewählt. Damit kann man alle genannten Felder bis auf niederfrequente Magnetfelder abschirmen. Diese niederfrequenten Störungen sind jedoch weniger problematisch, da einzelne Stromkurven zeitlich einen Bruchteil der Periodendauer dieser Störungen ausmachen und so nahezu als Gleichspannung betrachtet werden können.

Durch ein Gehäuse werden nur Störungen aus der Umgebung abgeschirmt. Kopplungen, die zwischen verschiedenen Schaltungsteilen auftreten, bleiben vom Gehäuse unberührt. Deshalb kann man auch mit einer perfekten Abschirmung keine perfekte Signalqualität erreichen. Man muss also trotz Gehäuse beim Aufbau der Schaltung auf mögliche Beeinflussungen zwischen

Schaltungsteilen achten bzw. auf kritische Schaltungsteile verzichten.

Batterieversorgung

Da heutzutage die Verwendung von Schaltnetzteilen sehr weit verbreitet ist, ist die Netzspannung von den Schaltspitzen der Schaltnetzteile stark überlagert. Diese Störungen werden in gedämpfter Form über ein typisches Labornetzgerät weitergegeben. Ein möglicher Ausweg ist die Verwendung einer Batterieversorgung. Batterien liefern eine sehr „glatte“ Gleichspannung ohne höherfrequente Störspannungen. Sie haben jedoch den Nachteil, dass mit der Betriebsdauer die abgegebene Spannung sinkt und so im Laufe einer Messreihe sich die Versorgungsspannung des Messobjekts ändert. Wenn die Versorgungsspannung unbemerkt unter die minimale Versorgungsspannung des Messobjekts sinkt, kann das schwer reproduzierbare Fehler verursachen.

Werden die Batterien mit in das Abschirmgehäuse eingebaut, fällt die Zuleitung der Versorgungsspannung weg. Damit hat man eine Leitungsdurchführung weniger und kann damit eine potentielle Störquelle ausschalten.

Da vor allem das ST10-Board eine relativ hohe Stromaufnahme von ca. 100 mA hat, wird im Laborbetrieb ein NiCd-Akku mit einer Kapazität von 1900 mAh verwendet. Die Spannung des Akkus von 7,2 V wird mittels Fixspannungsregler auf die benötigten 5 V heruntergeregelt. Beim ATmega32-Board wird eine herkömmliche 9 V Batterie verwendet, die im kleinen Aluminiumgehäuse untergebracht werden kann.

3.3. Beschreibung der Matlab-Skripte

Die Steuerung des Messsetups und das Auswerten der Messdaten erfolgte vorwiegend mit dem Mathematikprogramm *Matlab*[4]. Die Entscheidung fiel deshalb auf Matlab, da dieses Programm und die zugehörigen Toolboxen ein sehr mächtiges Werkzeug sind, bei dem man komplexe Berechnungen auf sehr schnelle Art und Weise umsetzen kann. Man kann daher Matlab für Rapid Prototyping verwenden, d. h. es ist möglich, eine Idee schnell umzusetzen und zu überprüfen, ob der Ansatz das gewünschte Ergebnis liefert. Mit der Instrument Control Toolbox ist es möglich, das Speicheroszilloskop über ein unkompliziertes Interface anzusteuern. Die Signal Processing Toolbox und die Statistics Toolbox bieten sehr mächtige Funktionen zur Auswertung der Messdaten an. Mit Hilfe von Matlab lassen sich Messdaten und Ergebnisse auf sehr einfache, aber auch visuell ansprechende Art als Diagramme darstellen. Eine Umsetzung der Software zur Aufnahme und Auswertung der Messdaten in einer Programmiersprache wie C würde einen ungleich höheren Aufwand bedeuten.

Jedoch hat Matlab nicht nur Vorteile: Die Instrument Control Toolbox weist eine mangelnde Stabilität auf. Bei der Ansteuerung des Oszilloskops verursacht dies Abstürze der kompletten Matlab-Umgebung. Es konnte festgestellt werden, dass es sich um einen Fehler in der Speicherverwaltung der Toolbox handelt. Mit entsprechenden Einstellungen der Größen der Sende- und Empfangsbuffer konnte dieses Problem weitestgehend behoben werden. Die Ansteuerung der seriellen Schnittstelle ist ebenfalls nicht sehr stabil. Es treten zwar keine Abstürze auf, aber unbegründete Timeouts beim Senden von Daten, was einen Abbruch des entsprechenden Skripts zur Folge hat. Weiterhin ist zu bemerken, dass es sowohl unter Windows als

[4] http://www.mathworks.com

auch unter GNU/Linux keine entsprechende Unterstützung der Ethernet-Schnittstelle in der Instrument Control Toolbox seitens Mathworks/Matlab bzw. des Oszilloskops/LeCroy gibt. Unter GNU/Linux wird selbst die GPIB-Schnittstelle nicht unterstützt, so dass Matlab dort für die Messwertaufnahme z. Z. nicht eingesetzt werden kann. Wenn viele Messdaten anfallen und dadurch sehr große Matrizen entstehen, kommt man schnell an die Grenzen der Speicherverwaltung von Matlab. Obwohl noch genügend Hauptspeicher vorhanden wäre, bricht Matlab oftmals mit einer Fehlermeldung das Skript ab.

Die im Rahmen dieser Diplomarbeit entstandenen Matlab-Skripte bauen auf Skripte auf, welche an der TU Graz am Institut für angewandte Informationsverarbeitung und Kommunikationstechnologie[5] (IAIK) entstanden sind. Unter anderem waren an der Entwicklung Stefan Mangard, Elisabeth Oswald, Christoph Herbst und Manfred Aigner beteiligt. Die Skripte wurden z. B. im Rahmen der Arbeiten [Man02],[Man04], [Osw03] sowie [AO00] entwickelt und verwendet. Nun soll auf die Grundstruktur der entstandenen Matlab-Skripte eingegangen werden und die Erweiterungen, welche im Rahmen dieser Arbeit entstanden sind, genauer besprochen werden. Abbildung 3.18 gibt einen generellen Überblick über den inneren Aufbau der DPA-Analysesoftware. Das Diagramm zeigt die wichtigsten Skripte und Klassen sowie deren Abhängigkeiten. Klassen kann man von Skripten dadurch unterscheiden, dass Klassennamen in Matlab immer mit einem „@ „beginnen. Die Pfeile zwischen den Klassen bzw. Skripten symbolisieren Abhängigkeiten (durchgezogene Linie) und Datenflüsse (gestrichelte Linie).

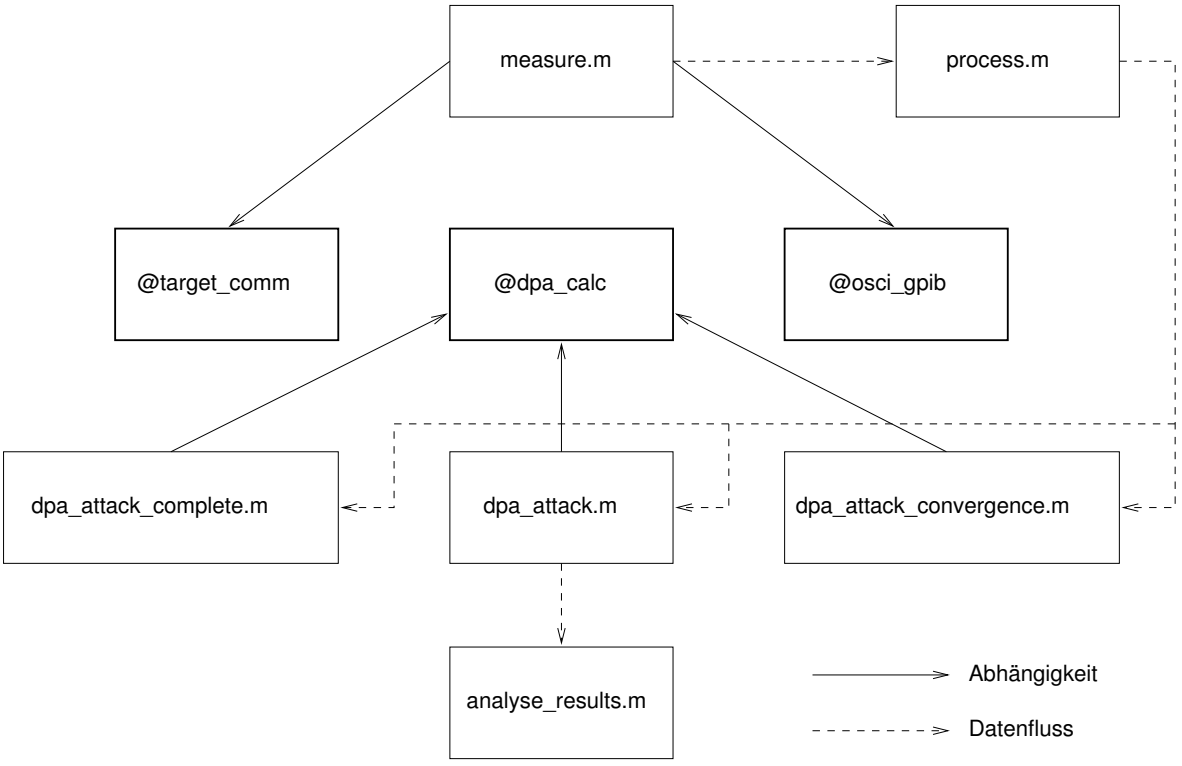

Abbildung 3.18: Klassendiagramm der Analysesoftware

[5] http://www.iaik.tu-graz.ac.at

3.3.1. Die Klasse @target_comm

Die Klasse `@target_comm` ist im Rahmen dieser Diplomarbeit entstanden. Sie abstrahiert die Kommunikationsschnittstelle zum Messobjekt. Die Klasse unterstützt die Kommunikation mit dem Messobjekt über die serielle Schnittstelle (ST10F168 und ATmega32) und die Kommunikation mit Chipkarten über die Microsoft PC/SC-Implementierung SCard.

Die Klasse ermöglicht auf sehr einfache Weise eine kryptographische Operation auf dem Messobjekt auszuführen:

```
1 target = target_comm('ATCOM');
2 plain = '0123456789abcdef';
3 cipher = encrypt(target, plain);
4 delete(target);
```

In der ersten Zeile wird ein Objekt der Klasse `@target_comm` angelegt, als Parameter muss das gewünschte Kommunikationsprotokoll angegeben werden. In diesem Beispiel wird die serielle Schnittstelle und ein Protokoll, welches an den AT-Modembefehlssatz angelehnt ist, ausgewählt. In der zweiten Zeile wird lediglich der Klartext als Zeichenkette der Variablen `plain` zugewiesen. In der Zeichenkette wird ein 64 Bit Block als hexadezimale Zahl ausgeschrieben. Die dritte Zeile führt den Befehl `encrypt` auf dem Messobjekt aus. Das Ergebnis der Verschlüsselungsoperation wird in der Variable `cipher` gespeichert. Der letzte Befehl trennt die Verbindung und löscht das Objekt der Klasse.

Zur Zeit werden zwei Kommunikationsprotokolle von der Klasse unterstützt, es besteht jedoch die Möglichkeit, die Klasse auf einfache Art und Weise an andere Zielplattformen anzupassen.

Kommunikation über die serielle Schnittstelle

Die serielle Schnittstelle wird mit 9600 Baud, 8 Datenbits, ohne Parität und einem Stopbit betrieben (9600 8N1). Die Verbindung mit dem Messobjekt erfolgt entweder über ein einfaches Nullmodemkabel oder über eine Verbindung mittels Lichtwellenleiter. Das Protokoll ist ähnlich dem AT-Modembefehlssatz aufgebaut und wurde auf dem ATmega32-Mikrocontroller und dem ST10F168-Mikrocontroller umgesetzt. Prinzipiell ist es auch möglich, dass man das Messobjekt mit einem Terminalprogramm ansteuert. Komfortabler ist jedoch das Ansprechen mittels Matlab-Klasse. Tabelle 3.5 zeigt die unterstützten Befehle. Die Parameter <key>, <plain> und <cipher> sind hexadezimale Zahlen mit einer Länge, die abhängig vom verwendeten Algorithmus bzw. dessen Blocklänge ist. Bei DES hat man eine Länge von 64 Bit (16 Zeichen) und bei AES beträgt die Länge 128 Bit (32 Zeichen).

Tritt bei der Kommunikation ein Fehler auf, so wird vom Mikrocontroller die Fehlermeldung `ERR` zurückgegeben. In so einem Fall kann mit dem Befehl `at` die Kommunikation in einen definierten Ausgangszustand gebracht werden.

Kommunikation über PC/SC-API

Für die Kommunikation mittels PC/SC wurde ein C-Programm geschrieben, das über die MEX-Schnittstelle von Matlab in die Software-Umgebung eingebunden wurde. Das Programm bedient sich der SCard-API, welche in Windows XP und Windows 2000 zum Lieferumfang gehört. Das C-Programm muss gegen die Bibliothek *winscard.lib* gelinkt werden. Unter Matlab kann man die MEX-Datei mit dem folgenden Befehl kompilieren:

Tabelle 3.5: Übersicht des Befehlssatzes

Befehl	Beschreibung
at	Ausgabe von 'OK'.
atr	Reset des Messobjekts und Ausgabe von 'OK'.
atk<key>	Setzen des verwendeten Schlüssels. Nur im Laborbetrieb nötig. Wenn der Schlüssel erfolgreich gesetzt wurde, wird er als Antwort zurückgesendet.
ate<plain>	Verschlüsseln des Klartexts <plain>. Als Antwort wird der Chiffretext zurückgesendet.
atd<cipher>	Entschlüsseln des Chiffretext <cipher>. Als Antwort wird der Plaintext zurückgesendet.
atm<mask>	Übertragung des Maskierungsblocks <mask> bei einer seitenkanalresistenten DES-Implementierung mit Maskierung.

```
mex mexPCSCapdu.c winscard.lib
```

Die Verwendung von PC/SC hat den großen Vorteil, dass jeder Chipkartenleser, für den es einen PC/SC-Treiber gibt, von der Software unterstützt wird. Die Schnittstelle auf der Matlab-Seite ist die gleiche wie bei der Kommunikation über die serielle Schnittstelle.

Die Kommunikation mit der Chipkarte erfolgt ausschließlich über das T=1 Protokoll. Die benötigte APDU[6] wird von der Matlab-Klasse erzeugt, das MEX-Programm übernimmt nur das Senden der APDU. Aus diesem Grund ist es einfach möglich, die Klasse `@target_comm` an gegebene Bedürfnisse anzupassen. Eine genaue Beschreibung der gesendeten APDUs kann man in Abschnitt 6 finden.

3.3.2. Die Klasse @osci_gpib

Die Klasse `@osci_gpib` war zu Beginn der Diplomarbeit schon vorhanden. Die Klasse abstrahiert die Kommunikation mit dem Oszilloskop und baut auf der Instrument Control Toolbox auf. Die Klasse beinhaltet Methoden für das Initialisieren und Konfigurieren des Speicheroszilloskops sowie die Möglichkeit, Messdaten vom Oszilloskop zum Laborrechner zu übertragen. Folgendes kleine Codefragment zeigt eine Kommunikation mit dem Oszilloskop, bei der ein Datensatz transferiert und in einem Vektor gespeichert wird:

```
1 osci = osci_gpib();
2 data = read(osci, 'C1');
3 delete(osci);
```

In der ersten Zeile wird die Verbindung aufgebaut und das Oszilloskop initialisiert. In der zweiten Zeile wird der Datensatz, der von Kanal eins gemessen wurde, ausgelesen und in der Variablen `data` gespeichert. In der letzten Zeile wird die Verbindung wieder abgebaut und das Objekt gelöscht.

[6] application protocol data unit

Diese Klasse verwendet die GPIB-Schnittstelle für die Kommunikation mit dem Oszilloskop. Die Verbindung über TCP/IP wird nicht verwendet, da das Oszilloskop für die Kommunikation über TCP/IP ein proprietäres Protokoll verwendet, das von der Instrument Control Toolbox nicht unterstützt wird. Der Hersteller des Oszilloskops liefert jedoch das ActiveX-Control ActiveDSO mit, über das man mit dem Oszilloskop kommunizieren kann. Die Anpassung dieser Softwareschnittstelle an Matlab wurde noch nicht vollständig vorgenommen. Über die TCP/IP-Verbindung über Ethernet kann man, wenn man vom Protokolloverhead absieht, mit bis zu 100 $^{\mathrm{MBit}}/_{\mathrm{s}}$ kommunizieren. Das wäre um den Faktor 12.5 schneller als über die GPIB-Schnittstelle, die eine Datenübertragungsrate von 8 $^{\mathrm{MBit}}/_{\mathrm{s}}$ zulässt. Bei längeren Messkurven, die beim verwendeten Oszilloskop bis zu 16 MB groß sein können, bedeutet das einen erheblichen Geschwindigkeitsvorteil.

3.3.3. Die Klasse @dpa_calc

Die Klasse `@dpa_calc` ist sozusagen das Herzstück der Analysesoftware. Sie beinhaltet Implementierungen der statistischen Verfahren, welche in Abschnitt 2.3.3 bereits besprochen wurden. Die Klasse unterstützt also die Datenanalyse mit der Methode nach Kocher und mit der Korrelationsmethode. Mit einer speziellen Methode werden Messkurven zum Objekt hinzugefügt. Mit einer anderen Methode werden die Analyse der Daten ausgeführt und die Differenzkurven bzw. die Korrelationskurven ausgegeben. Diese Vorgehensweise ermöglicht es, nach einer (noch) nicht erfolgreichen Analyse der Kurven weitere Messkurven zum Objekt hinzuzufügen. Die Messkurven werden nicht als Rohdaten in der Klasse gespeichert, sondern es erfolgt bereits beim Hinzufügen der einzelnen Kurven eine Summenbildung.

Die Klasse war zu Beginn der Diplomarbeit weitestgehend vorhanden, sie wurde jedoch im Lauf der Arbeit verbessert. Die ursprüngliche Klasse hat nur das Hinzufügen von einzelnen Messkurven zugelassen. In Matlab ist es jedoch von Vorteil, wenn man auf Matrixoperationen zurückgreift, da diese sehr schnell ausgeführt werden können. In der verbesserten Version werden Matrixoperationen verwendet, womit es möglich ist, mehrere Messkurven mit einem Methodenaufruf hinzuzufügen. Das erhöht, wenn man die Messkurven aus Dateien liest, die Verarbeitungsgeschwindigkeit um den Faktor zehn.

3.3.4. Realisierung der Auswahlfunktionen von DES und AES

Sehr wichtig für DPA-Seitenkanalattacken sind Auswahlfunktionen für die anzugreifenden Algorithmen. In Anschnitt 2.3.3 werden zwei mögliche Auswahlfunktionen jeweils für den DES und den AES vorgestellt. Hier soll kurz erläutert werden, wie diese Auswahlfunktionen in Matlab realisiert werden. Die Schnittstelle zu den Auswahlfunktionen besteht jeweils aus einer einzigen Matlab-Funktion, die bei DES eine MEX-Routine aufruft und bei AES eine Berechnung startet, welche vollständig in Matlab implementiert ist. Dieser Weg wurde gewählt, da beim DES Rechenoperationen auf einzelne Bits angewendet werden müssen und das in der Matlab-Skriptsprache sehr umständlich ist. Beim AES müssen nur Operationen auf Bytes ausgeführt werden, das ist in Matlab einfach umzusetzen.

Die Funktion ermöglicht das Berechnen der *D-Matrix* für einen Satz von Klartexten. In dieser D-Matrix wird das Ergebnis der Auswahlfunktion $b = \mathrm{D}(P, k_n)$ für jede mögliche Schlüsselhypothese k und mehrere Klartexte P gespeichert. Dabei steht jede Spalte für eine Schlüsselhypothese und jede Zeile für einen Klartext. Abhängig vom Algorithmus kann man

festlegen, welches Bit und welche S-Box man angreifen will. Des weiteren kann man festlegen ob eine D-Matrix für eine „Single Bit DPA“ oder eine „Multiple Bit DPA“ berechnet werden soll. Das Codefragment,

```
1 plaintexts = ['0123456789abcdef'; 'aaaabbbbcccc1234'];
2 dmatrix = DES_compute_dmatrix(plaintexts, 'corr', false, 1, 1);
```

berechnet die D-Matrix für zwei Klartextblöcke einer DES-Verschlüsselung. Die Klartextblöcke sind in `plaintexts` abgelegt. Da die D-Matrix bei der Implementierung in Matlab unterschiedlich für die Kocher- und die Korrelationsmethode definiert ist, muss hier angegeben werden, welche Methode für die Analyse der Messdaten verwendet wird. Die letzten drei Parameter bedeuten, dass das erste Bit (Single Bit DPA) der ersten S-Box angegriffen werden soll.

Die äquivalente Matlab-Funktion für den AES wird mit `AES_compute_dmatrix` aufgerufen.

3.3.5. Workflow

In der ursprünglichen Lösung wurde die Analyse von einem zentralen Skript gesteuert, dabei waren das Messen und die Auswertung eng miteinander verschränkt. Im neuen Ansatz, der im Rahmen dieser Arbeit entstanden ist, erfolgt die DPA-Attacke in mehreren Etappen, damit im Zuge einer Untersuchung die Möglichkeit besteht, die Messdaten flexibel verschiedenen Aufbereitungsmethoden und Analysemethoden zu unterziehen. Abbildung 3.19 zeigt den Arbeitsablauf einer solchen DPA-Untersuchung.

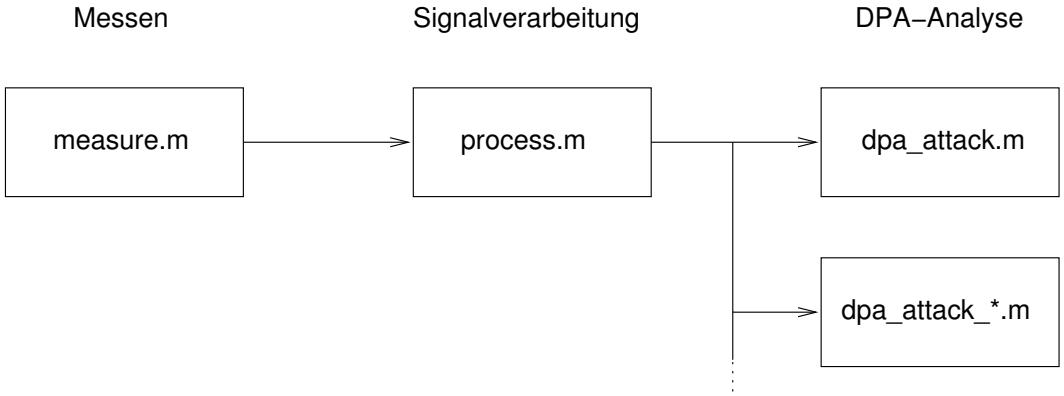

Abbildung 3.19: Workflow einer Seitenkanalattacke

measure.m

Das Skript `measure.m` führt die eigentliche Messung durch und speichert die Messdaten in Binärdateien ab. Dabei bedient sich das Skript der Klassen `@target_comm` und `@osci_gpib`. Zusätzlich zum reinen Speichern der Messdaten wird vom Skript ein Unterverzeichnis angelegt, in das die Messdaten und Informationen zu den Messdaten abgelegt werden. Die Informationen zu den Messdaten sind in lesbarer Form als Textdatei vorhanden, damit die Rahmenbedingungen der Messung später nachvollzogen werden können. Des Weiteren werden Parameter für andere Matlab-Skripte in Form von mat-Dateien abgelegt. Das hat den

Vorteil, das man z. B. bei einer Analyse der Daten dem Analyseskript nicht mitteilen muss, um welchen kryptographischen Algorithmus es sich handelt.

process.m

Das Skript `process.m` bereitet die rohen Messdaten auf, damit man sie im darauf folgenden Schritt analysieren kann. Folgende vier Aufgaben übernimmt das Skript:

- Es wird ein weiteres Unterverzeichnis angelegt, in dem die aufbereiteten Daten gespeichert werden. Des Weiteren wird eine Textdatei angelegt, in die die Einstellungen des `process.m`-Skriptes geschrieben werden. Somit kann man später nachvollziehen, welche Parameter für die Aufbereitung der Daten verwendet wurden.

- Die zeitliche Verschiebung (Phasenjitter) wird durch Pattern-Matching ausgeglichen. Dieses Feature hat bei der Messung von Mikrocontrollern keine große Bedeutung, da man den Startzeitpunkt der Messung genau festlegen kann. Bei Chipkarten kann man den Startzeitpunkt jedoch nicht genau bestimmen, weshalb die zeitliche Verschiebung ausgeglichen werden muss.

- Wenn man eine ungefähre Vorstellung hat, zu welchen Zeitpunkt die angegriffenen Bits berechnet werden, kann man die Kurven „zurechtschneiden“ und erreicht damit eine Reduktion der Daten, was sich in einer Geschwindigkeitssteigerung bei der Analyse äußert.

- Da die Messdaten mit einer relativ hohen Samplingrate aufgezeichnet werden, um die Zuverlässigkeit des Pattern-Matching zu garantieren, kann man nach der Korrektur der zeitlichen Verschiebung die Datenmenge durch Downsampling reduzieren.

dpa_attack*.m

Die eigentliche Auswertung der Daten erfolgt in den `dpa_attack*.m`-Skripten. Jedes dieser Skripte hat eine spezielle Aufgabe, die in diesem Abschnitt beschrieben werden soll.

- `dpa_attack.m` – `analyse_results.m`
  Dieses Skript berechnet die Differenzkurven oder die Korrelationskurven für einen Teilschlüssel. Es werden also 6 Bit eines DES-Schlüssels oder 8 Bit eines AES-Schlüssels ermittelt. Die vier wahrscheinlichsten der berechneten Kurven werden mit dem Skript `analyse_results.m` als Plots ausgegeben. Das Skript `analyse_results.m` sucht dabei nach den vier Kurven mit den höchsten Spitzenwerten. Der Benutzer der Software kann so visuell überprüfen, ob sich die Spitzen wirklich eindeutig von den anderen Amplitudenwerten abheben.

- `dpa_attack_complete.m`
  Dieses Skript kann aus einem Datensatz einen kompletten Schlüssel extrahieren. Es werden dabei keine Plots ausgegeben, sondern nur der gefundene Schlüssel. Beim DES müssen dafür die Teilschlüssel aller acht S-Boxen ermittelt werden. Danach werden die fehlenden acht Bits mit einer Brute-Force-Attacke gesucht. Beim AES kann ein gesamter 128 Bit Schlüssel mit einer 16-fachen Analyse der Daten gefunden werden.

- `dpa_attack_convergence.m`
  Mit diesem Skript kann man feststellen, bei wie vielen Messungen das Ergebnis der DPA-Analyse konvergiert, d. h. wie viele Messungen man benötigt, um den geheimen Schlüssel zu finden. Das Skript ermittelt dabei für jeden Teilschlüssel und für jedes mögliche Zielbit die Anzahl der Messungen, die nötig sind, bis das Ergebnis konvergiert.

Kapitel 4.

Atmel ATmega32 – Messaufbau und Ergebnisse

In diesem Kapitel soll der Messaufbau basierend auf dem Atmel[1] ATmega32 Mikrocontroller beschrieben werden. Diese Mikrocontrollerarchitektur wurde gewählt, da ein hochentwickelter C-Compiler dafür frei verfügbar ist. Des weiteren war ein Entscheidungsgrund, dass man ohne große externe Beschaltung einen voll funktionsfähigen und auch leistungsfähigen Rechner erhält. Durch die geringe externe Beschaltung kann man Störungen, die von anderen Schaltungsteilen hervorgerufen werden, stark reduzieren.

4.1. Prozessorarchitektur und Entwicklungstools

Beim ATmega32 Mikrocontroller von Atmel handelt es sich um einen 8 Bit RISC-Controller, der über eine Rechenleistung von bis zu einem MIPS pro MHz verfügt. Die meisten Befehle des Mikrocontrollers haben eine Ausführungszeit von nur einem Takt. Der Mikrocontroller hat ein voll statisches Design, das bedeutet, man kann die Taktfrequenz bis zum Stillstand absenken (0–16 MHz). Der Befehlssatz besteht aus 131 Befehlen. Es stehen 32 acht Bit breite Register zur Verfügung, die sich im Prozessorkern befinden und direkt mit der ALU verbunden sind. Eine Multipliziereinheit, mit der man zwei acht Bit Zahlen in zwei Takten multiplizieren kann, ist ebenfalls vorhanden.

Neben dem Registersatz verfügt der Mikrocontroller über verschiedene getrennte Speicherbereiche. Als erstes sei der Flash-Speicher erwähnt, welcher für die Speicherung der Software zuständig ist und eine Größe von 32 KB hat. Ein SRAM mit einer Größe von 2 KB zur Speicherung von Daten während der Laufzeit ist ebenfalls vorhanden. Diese zwei Speicherbereiche haben getrennte Adressräume. Zusätzlich gibt es noch ein EEPROM mit einer Größe von 1 KB zur Speicherung von Konfigurationsparametern, die nach dem Abschalten der Versorgungsspannung erhalten bleiben sollen.

Der ATmega32 besitzt eine Vielzahl von on-Chip Peripherie, wie z. B. Zähl- und Timereinheiten, synchrone und asynchrone serielle Schnittstellen sowie einen analog-digital-Wandler (ADC). Die Verbindung mit der Außenwelt erfolgt über 32 I/O-Leitungen, die entweder als digitale I/O oder als Schnittstellen für die on-Chip Peripherie konfiguriert werden können.

Abbildung 4.1 zeigt das Blockschaltbild des ATmega32, in dem die einzelnen Peripheriekomponenten und die Verbindungen zum CPU-Kern zu sehen sind. Dieses Blockschaltbild ist dem Datenblatt [ATM04] entnommen, in dem eine detaillierte Beschreibung aller Komponenten des Mikrocontrollers zu finden ist.

[1] http://www.atmel.com

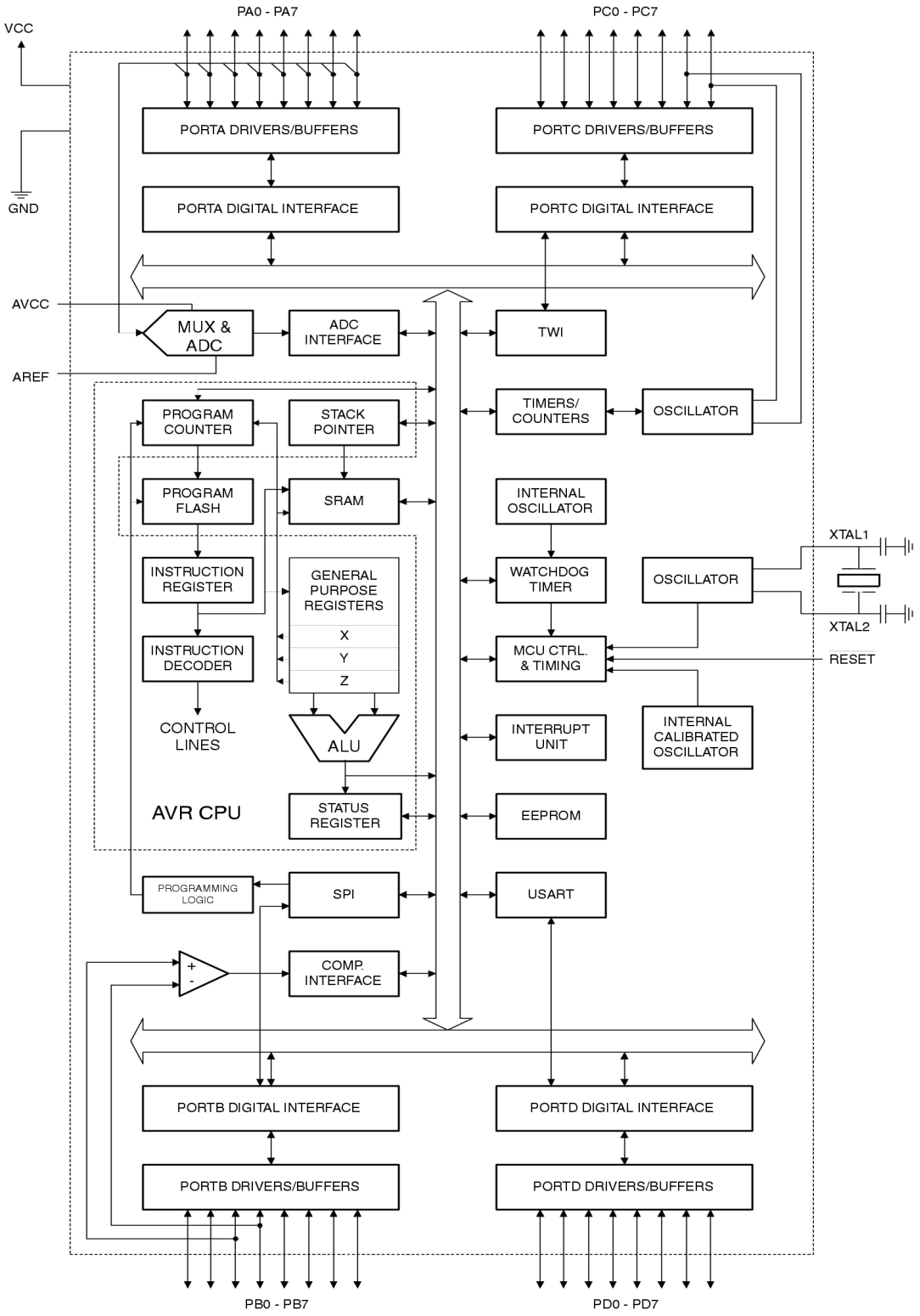

Abbildung 4.1: Blockschaltbild des ATmega32

Entwicklungstools

Die Software für den ATmega32 Mikrocontroller wurde mit Hilfe des *avr-gcc* C-Compilers entwickelt. Das ist eine spezielle Version des GNU C-Compilers, die an die Prozessorarchitektur von Atmel angepasst wurde. Gängige GNU/Linux Distributionen stellen entsprechende Pakete zur Verfügung. Für Microsoft Windows gibt es das Komplettpaket Winavr[2].

Für die Programmierung des Mikrocontrollers benötigt man zusätzlich zum Compiler die Programmbibliothek *AVR-Libc*[3], in der Funktionen zum komfortablen Ansteuern der on-Chip Peripherie und verschiedene Funktionen aus der Standard-C-Bibliothek implementiert sind. Eine detaillierte Dokumentation der AVR-Libc kann man unter [AL05] finden.

Für das Einspielen der Software verfügt der Mikrocontroller über eine spezielle synchrone, serielle Schnittstelle. Diese Schnittstelle ermöglicht das Programmieren des Flash-Speichers innerhalb einer Schaltung, ohne dass der Mikrocontroller aus der Schaltung entnommen werden muss. Dieses Verfahren wird „*in system programming*“ (ISP) genannt. Um den Flash-Speicher über diese Schnittstelle zu programmieren, benötigt man einen ISP-Adapter, der im einfachsten Fall an die parallele Schnittstelle des PCs angeschlossen und mit entsprechender Software angesteuert wird. Es gibt verschiedene Ausführungen von ISP-Adaptern und ISP-Software. In der Arbeit kam ein STK200 [WIK05] kompatibler ISP-Adapter und die Programmiersoftware *uisp*[4] unter GNU/Linux zum Einsatz.

Als Abrundung der Entwicklungsumgebung kann man mit *make* alle Arbeitsschritte der Softwareentwicklung bis hin zum Programmieren des Flash-Speichers automatisieren.

Die Software für den Atmel Mikrocontroller wurde unter Debain GNU/Linux entwickelt. In Tabelle 4.1 sind die verwendeten Debian-Pakete und die Versionsnummer aufgeführt.

Tabelle 4.1: Debian-Pakete, mit deren Hilfe die Atmel-Software entwickelt wurde

Bezeichnung	Version
binutils-avr	2.15-1
gcc-avr	3.4.3-1
avr-libc	1.0.5-1
uisp	20040311-1

4.2. Schaltung

Die Schaltung des Messobjekts ist gewollt möglichst einfach gehalten, damit das Messergebnis nicht durch unnötige Störungen, die von zusätzlichen Schaltungsteilen ausgehen könnten, beeinflusst wird. Abbildung 4.2 zeigt den Schaltplan, der in drei Blöcke gegliedert werden kann. Als ersten und wichtigsten Block kann man den Mikrocontroller mit minimaler externer Beschaltung sehen. In den Versorgungsleitungen kann man die Shunt-Widerstände erkennen, die aus einer Parallelschaltung von zwei 10 Ω Widerständen gebildet werden, was in einem

[2] http://winavr.sourceforge.net
[3] http://www.nongnu.org/avr-libc
[4] http://www.nongnu.org/uisp

Widerstandswert von 5 Ω resultiert. Die Schaltung besitzt zwei Widerstandspaare, was das Messen des Spannungsabfalls gegen Masse und gegen die Versorgungsspannung erlaubt.

Der zweite Schaltungsteil ist die Ansteuerung der optischen Bauelemente für die Kommunikation über die Lichtwellenleiter (LWL). Die gleiche Schaltung, erweitert um eine Pegelanpassung von 5V CMOS Pegel auf RS232 Pegel, ist auf der Seite des PCs vorhanden. Die Kommunikation erfolgt über zwei 2,2 mm dicke Kunststoff-LWL, die den Vorteil haben, dass sie leicht zu verarbeiten sind.

Der dritte Schaltungsteil ist eine Spannungsstabilisierung auf fünf Volt mittels Fixspannungsregler. Die Platine muss also mit mindesten sieben Volt versorgt werden. Im Labor kommt eine 9 V Batterie und das Labornetzgerät zum Einsatz.

Diese endgültige Version ist eine Verbesserung eines Versuchsaufbaus basierend auf dem ATmega16 Mikrocontroller, der bis auf den verfügbaren Speicher mit dem ATmega32 identisch ist. Beim ersten Versuchsaufbau entstanden vor allem bei der Pegelwandlung der seriellen Schnittstelle durch den MAX232 Pegelwandler Probleme. Durch die Einführung der optischen Verbindung wurden diese beseitigt.

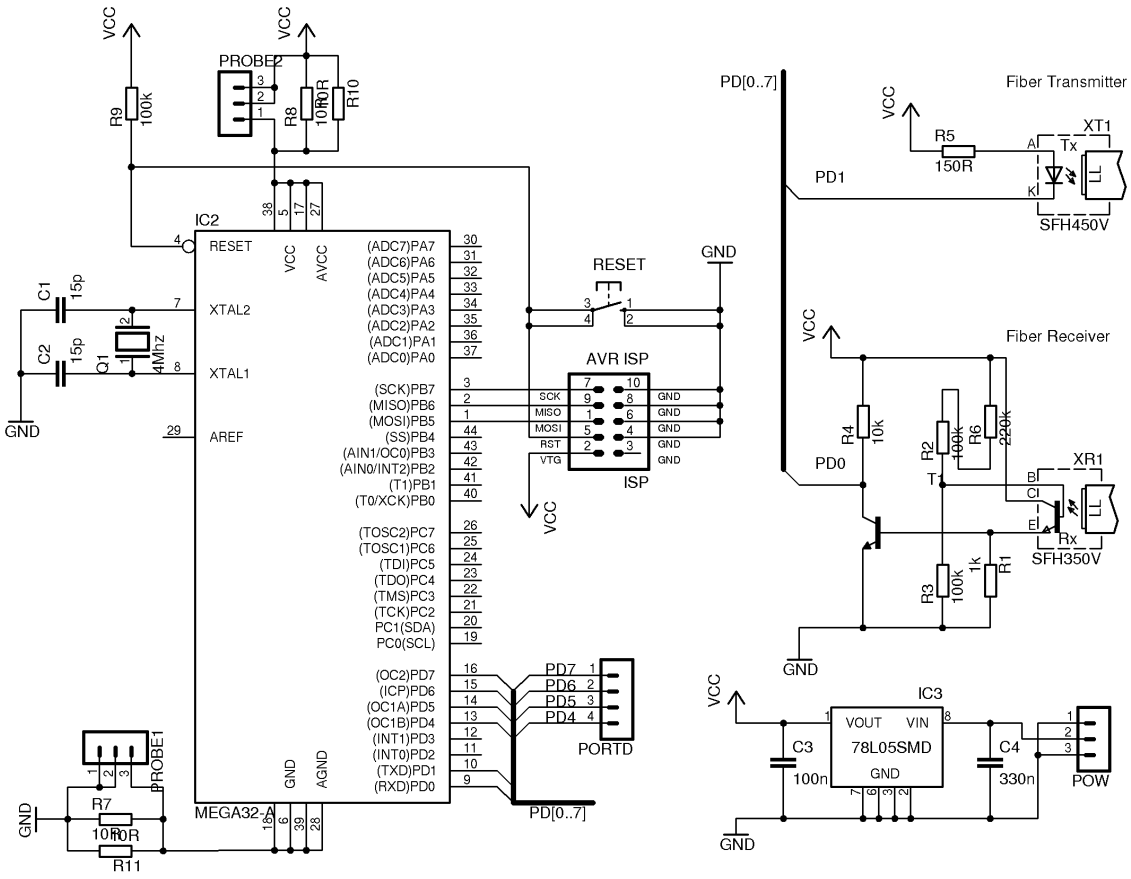

Abbildung 4.2: Schaltplan des ATmega Messobjekts

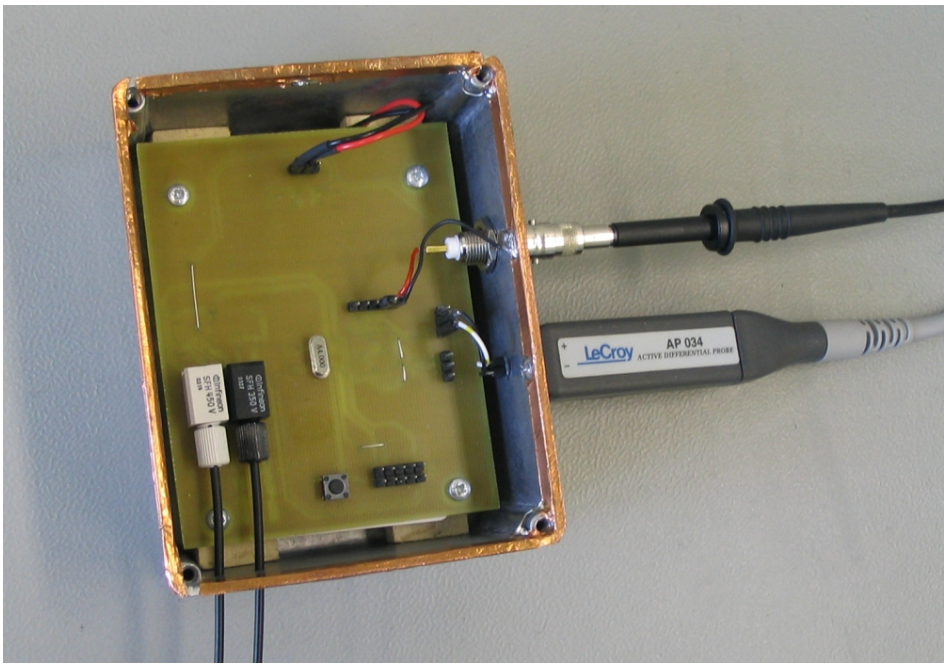

Abbildung 4.3: Foto des ATmega32 Messobjekts

4.3. Analysierte Software

Auf den ATmega32 Mikrocontroller wurde eine C-Implementierung des DES-Algorithmus (siehe [NIS99] und [MvOV96, S. 250ff]) portiert. Die Implementierung ist bei Siemens CT IC 3 entwickelt worden und verwendet verschiedene Optimierungsverfahren zur Geschwindigkeitssteigerung. Gegenmaßnahmen gegen DPA-Attacken wurden in dieser Implementierung nicht vorgesehen.

4.4. Messergebnisse

4.4.1. Charakterisierung der Stromaufnahme

Das Diagramm in Abbildung 4.4 zeigt die Stromaufnahme des ATmega32 Mikrocontrollers während der Berechnung der 16 DES-Runden. Die Daten für dieses Diagramm stammen direkt vom Oszilloskop und wurden mit einer Abtastrate von 250 $^{\mathrm{MS}}/_{\mathrm{s}}$ aufgezeichnet. Die Messdaten wurden nicht nachbearbeitet. Die 16 Runden der DES-Operation kann man deutlich in der Stromkurve erkennen.

Das Spektrum der Stromaufnahme ist in Abbildung 4.5 dargestellt. Die Amplitudenwerte in diesem Diagramm sind nicht auf einen physikalischen Wert normiert. Der dominierende Frequenzanteil neben der Gleichspannung ist die zweite Oberwelle des Prozessortaktes von 4 MHz. In diesem Frequenzbereich ist auch der größte Anteil der Seitenkanalinformation enthalten.

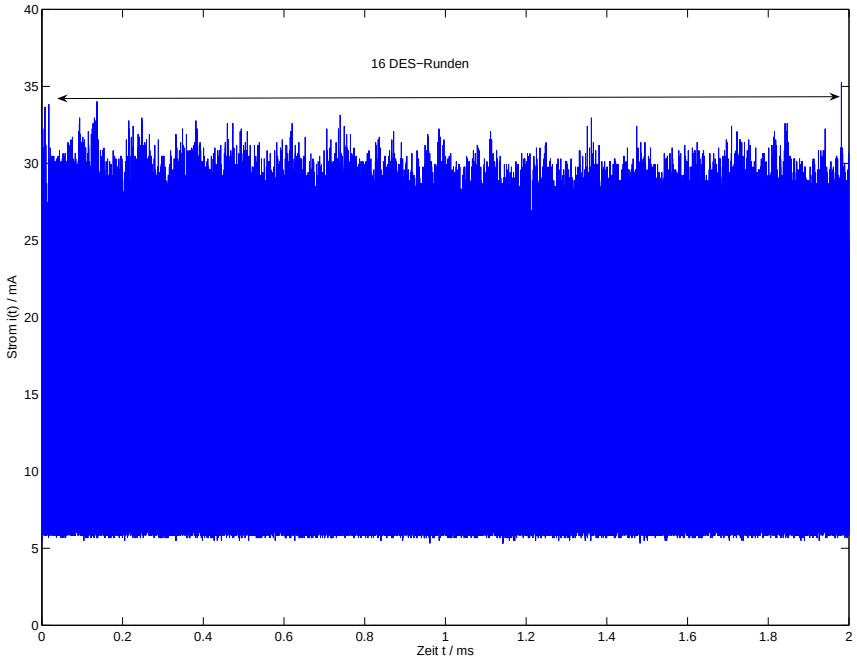

Abbildung 4.4: Stromaufnahme des ATmega32-Mikrocontrollers bei der Berechnung einer DES-Operation

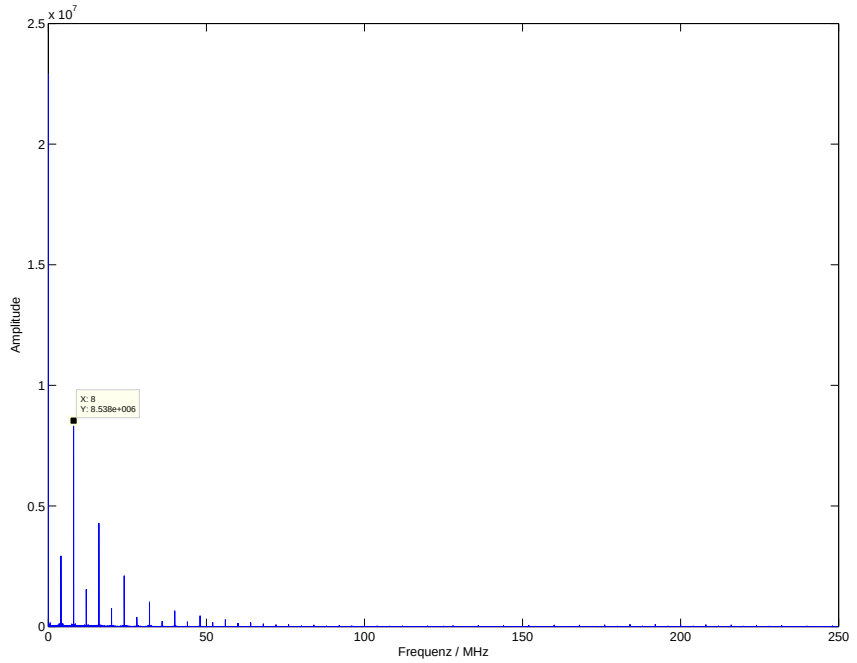

Abbildung 4.5: Spektrum der Stromaufnahme des ATmega32. Neben dem Gleichspannungsanteil ist die zweite Oberwelle bei 8 MHz deutlich ausgeprägt.

4.4.2. Angriff auf eine exemplarische S-Box

Nun soll die erste S-Box und das erste Target-Bit mit verschiedenen Methoden angegriffen werden. Der richtige Teilschlüssel für die angegriffene Struktur hat den Wert 2. Dieser Schlüssel konnte auch bei allen hier gezeigten Angriffen gefunden werden. Abbildung 4.6 zeigt die Ergebnisse zweier DPA-Attacken mittels Korrelationsmethode. In der oberen Hälfte der Abbildung sind 200 Messungen ($N = 200$) und in der unteren Hälfte 1000 Messungen ($N = 1000$) in die Analyse eingeflossen. Die Messdaten wurden mit einer Abtastfrequenz von 2,5 $^{\mathrm{GS}}/_{\mathrm{s}}$ aufgezeichnet und einem Downsampling mit dem Faktor 50 unterzogen. Die Abtastfrequenz von 2,5 $^{\mathrm{GS}}/_{\mathrm{s}}$ entspricht einem Oversampling von 625. Die Abtastfrequenz hat sich aus einer fest eingestellten Länge des Oszilloskop-Sampling-Speichers beim Einstellen des gewünschten Zeitabschnitts ergeben. Das Oversampling reduziert den Phasenjitter, die gemessenen Daten können jedoch vor der Analyse gefiltert werden und die Datenmenge mittels Downsampling reduziert werden.

In Abbildung 4.6 links ist die jeweils wahrscheinlichste Korrelationskurve dargestellt. Rechts werden für jede der 64 möglichen Schlüsselhypothesen die Absolutwerte der Spitzenwerte der einzelnen Korrelationskurven dargestellt. Als Vergleich dazu ist in Abbildung 4.7 das Ergebnis der Attacke mit gleichen Messdaten, jedoch unter Verwendung der Kochermethode, dargestellt. Man sieht, dass die Differenzkurven nicht auf ± 1 normiert sind.

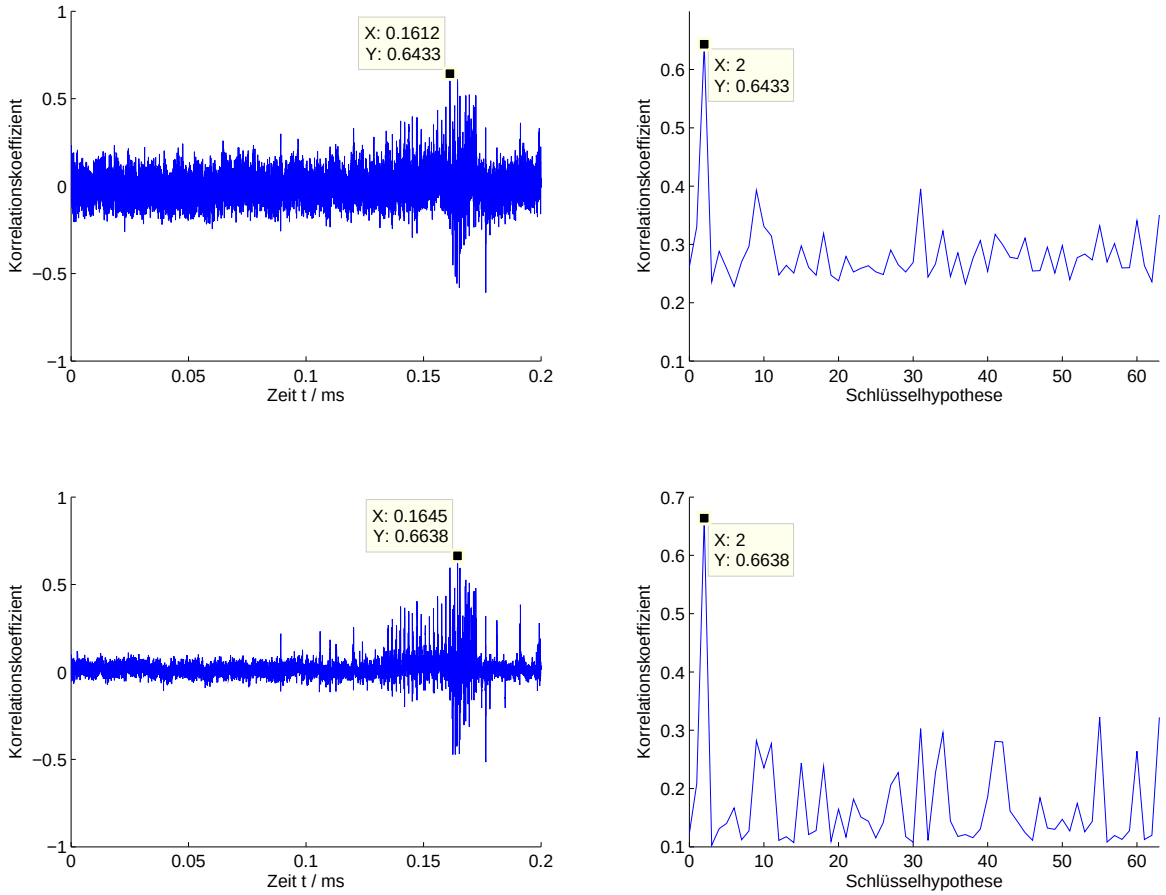

Abbildung 4.6: Resultat eines DPA-Angriffs mittels Korrelationsmethode auf die erste S-Box und das erste Target-Bit. Der zugehörige richtige Teilschlüssel ist 2. In der oberen Hälfte wurden 200 Messkurven ($N = 200$) und in der unteren Hälfte 1000 Messkurven ($N = 1000$) analysiert. Links sieht man die wahrscheinlichste Korrelationskurve und rechts die Absolutwerte der Maxima aller 64 Korrelationskurven.

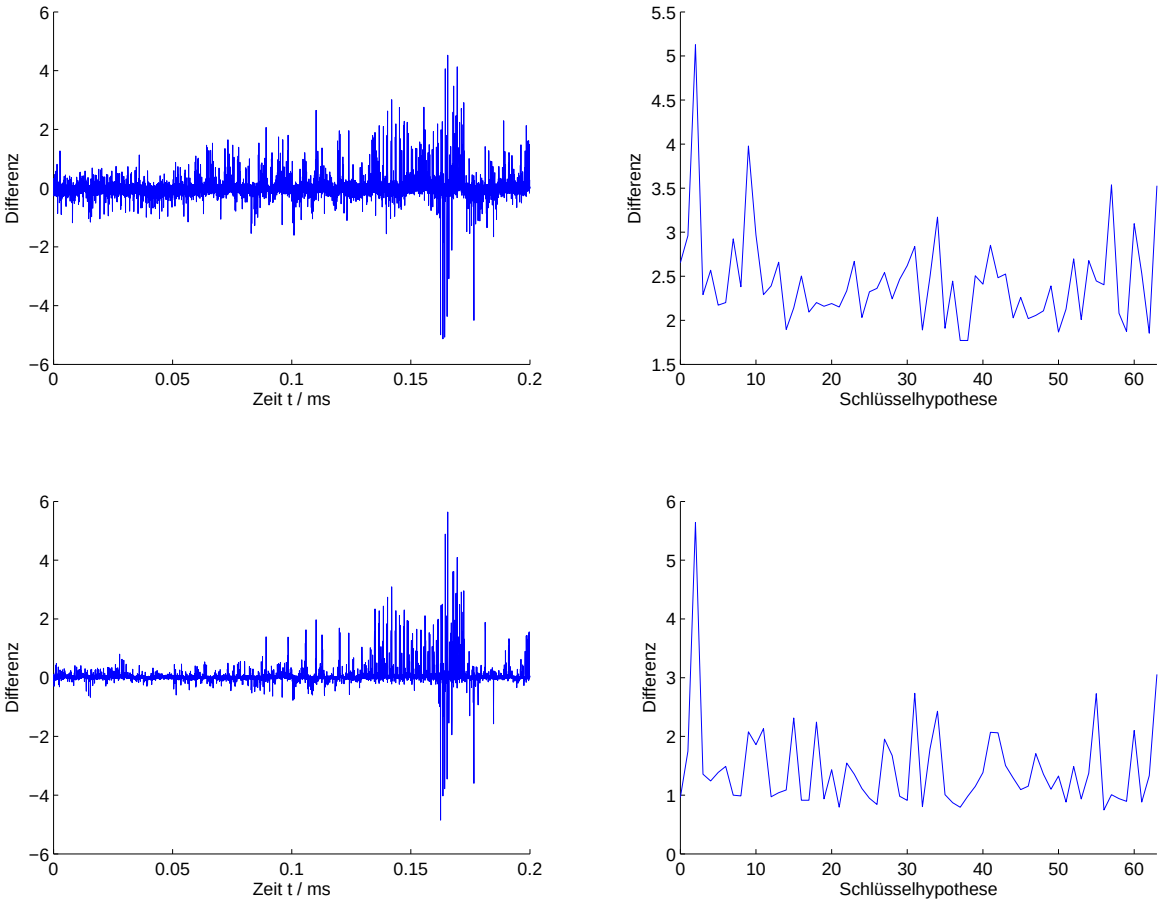

Abbildung 4.7: Resultat eines DPA-Angriffs mittels Kochermethode auf die erste S-Box und das erste Target-Bit. Der zugehörige richtige Teilschlüssel ist der Wert 2. In der oberen Hälfte wurden 200 Messkurven ($N = 200$) und in der unteren Hälfte 1000 Messkurven ($N = 1000$) analysiert. Links sieht man die wahrscheinlichste Korrelationskurve und rechts die Absolutwerte der Maxima aller 64 Korrelationskurven.

4.4.3. Anzahl der benötigten Messungen

In diesem Abschnitt soll gezeigt werden, wie viele Messungen im Schnitt benötigt werden, damit man bei diesem Messobjekt einen erfolgreichen DPA-Angriff durchführen kann. Dabei wird gegenübergestellt, wie viele Messungen bei der Verwendung eines Gehäuses und wie viele Messungen ohne Gehäuse benötigt werden. In Abbildung 4.8 wurde der Schlüssel `0x0123456789abcdef` verwendet. Im rechten Diagramm wurde das Messobjekt im Gehäuse, wie es in Abbildung 4.3 gezeigt wird, untergebracht. Im linken Diagramm wurde auf das Gehäuse verzichtet. Um das Minimum der benötigten Messkurven zu ermitteln, wurden einmalig 350 Messkurven aufgenommen und danach mehreren DPA-Analysen hintereinander unterzogen, wobei die Anzahl der ausgewerteten Kurven kontinuierlich erhöht wurde. Wie man in den Diagrammen sehen kann, „konvergieren“ die Ergebnisse der Analysen bei deutlich kleinerer Anzahl von Messungen zum richtigen bekannten Wert. Auf der Ordinate der Diagramme ist dabei die Anzahl der Messungen aufgetragen, auf der Abszisse die jeweilige S-Box, zu der vier Target-Bits gehören. Abbildung 4.9 stellt das gleiche Ergebnis dar, mit dem Unterschied, dass als Schlüssel `0xaffe089182588bbb` verwendet wurde. In Abbildung 4.8 konnte tatsächlich die mittlere Anzahl der benötigten Messungen, bei der Verwendung eines Gehäuses, verringert werden. In Abbildung 4.9 ist hingegen die benötigte Anzahl im Mittel sogar etwas größer. Dieser hohe Mittelwert ist auf den „Ausreißer“ bei S-Box 5 und Target-Bit 1 zurückzuführen. Aus den Diagrammen kann man auch erkennen, das die Anzahl der benötigten Messungen sehr von der angegriffenen S-Box, vom angegriffenen Target-Bit und vom Schlüssel abhängen.

In Abbildung 4.10 soll gezeigt werden, wie sich verschiedene Downsampling-Faktoren auf die Anzahl der benötigten Messungen auswirken. Es kann gezeigt werden, dass bei Einhaltung des Abtasttheorems ein größerer Faktor die Anzahl der benötigten Messungen reduziert. Das ist auf die Reduktion des Rauschens und dem resultierenden besseren SNR zurückzuführen. Die zugrunde liegenden Messdaten wurden bei einer Abtastfrequenz von 2,5 $\mathrm{GS/s}$ aufgezeichnet, das entspricht einem 625-fachen Oversampling. Im linken Teil des Diagramms wurde der Downsampling Faktor 50 verwendet, im rechten Teil wurde der Faktor 100 verwendet. Obwohl man die Daten nach dem Messen durch Downsampling so stark reduziert, ist es wichtig, zunächst mit hohen Abtastraten zu arbeiten, damit der Phasenjitter möglichst klein bleibt.

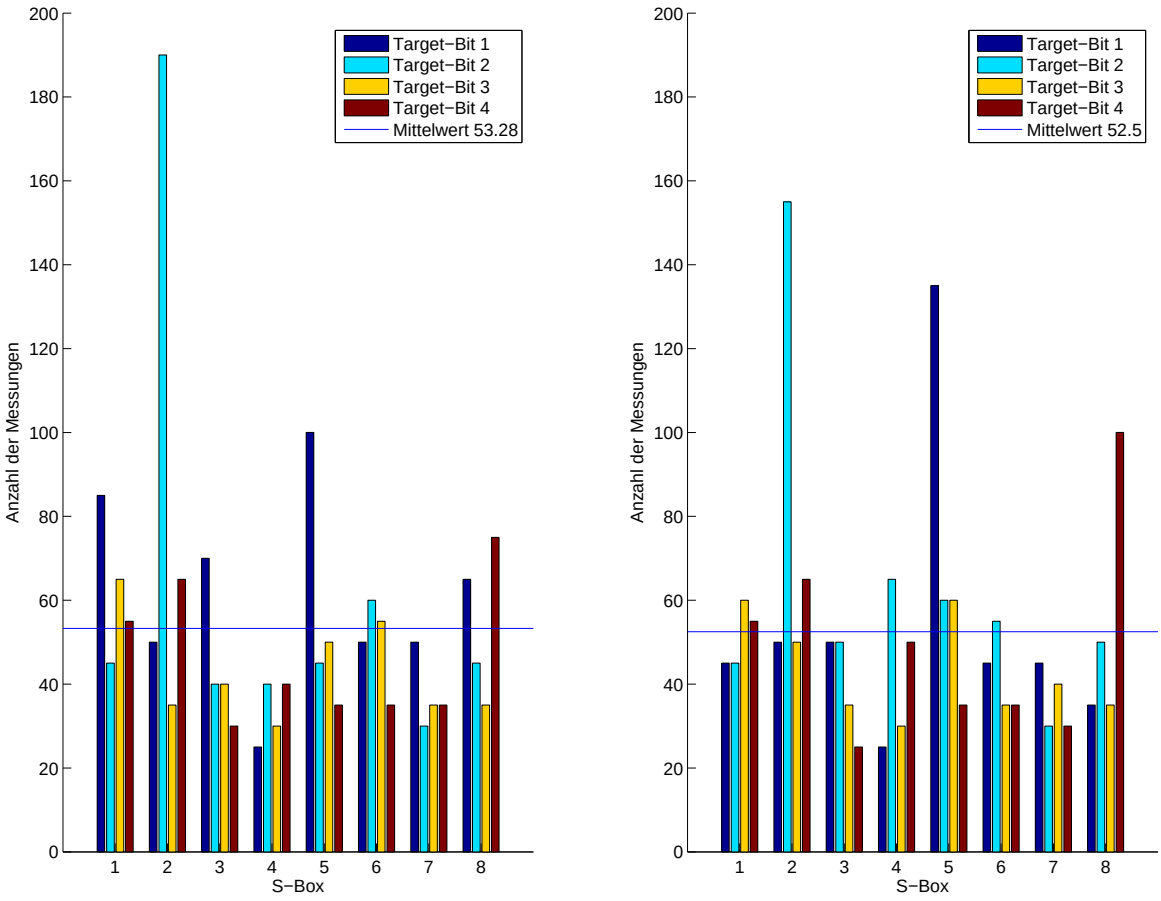

Abbildung 4.8: Anzahl der benötigten Messungen für eine erfolgreiche DPA-Attacke ohne Gehäuse (links) und mit Gehäuse (rechts). Verwendeter Schlüssel ist `0x0123456789abcdef`.

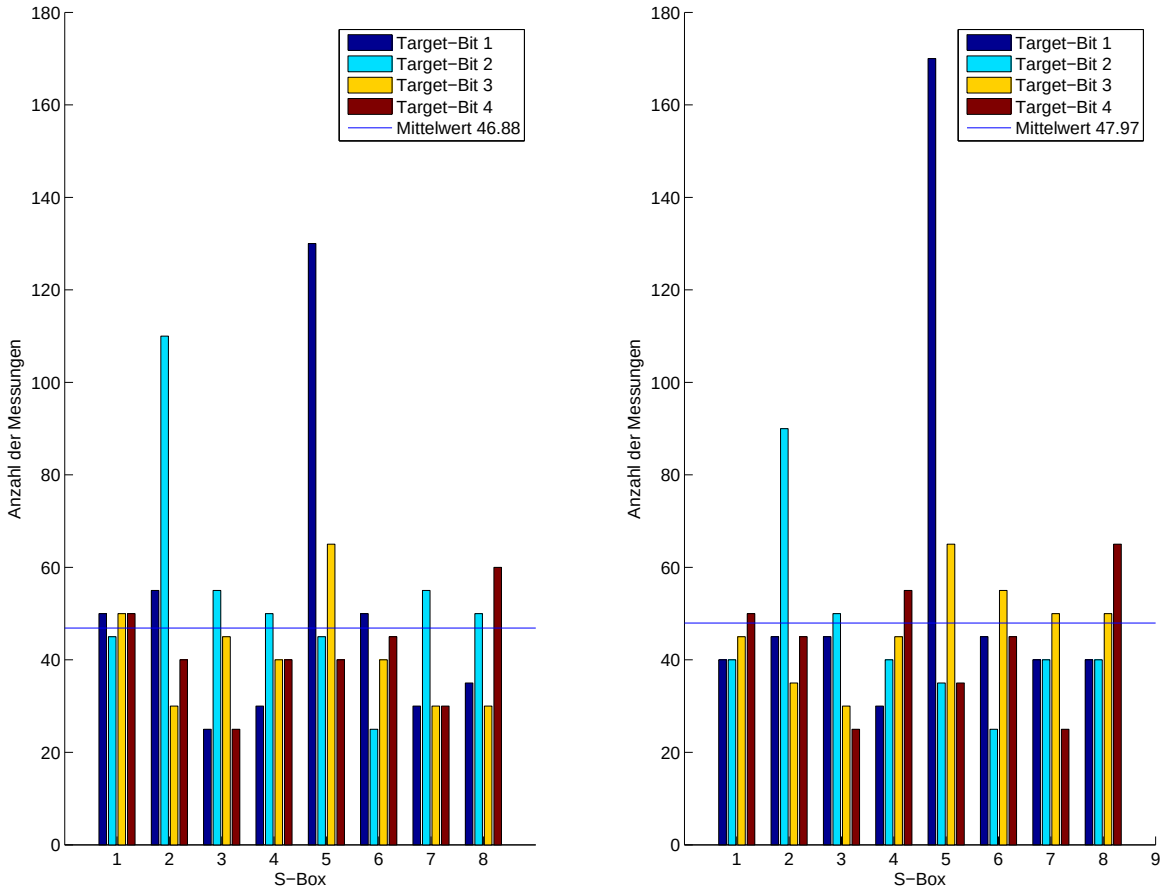

Abbildung 4.9: Anzahl der benötigten Messungen für eine erfolgreiche DPA-Attacke ohne Gehäuse (links) und mit Gehäuse (rechts). Verwendeter Schlüssel ist `0xaffe089182588bbb`.

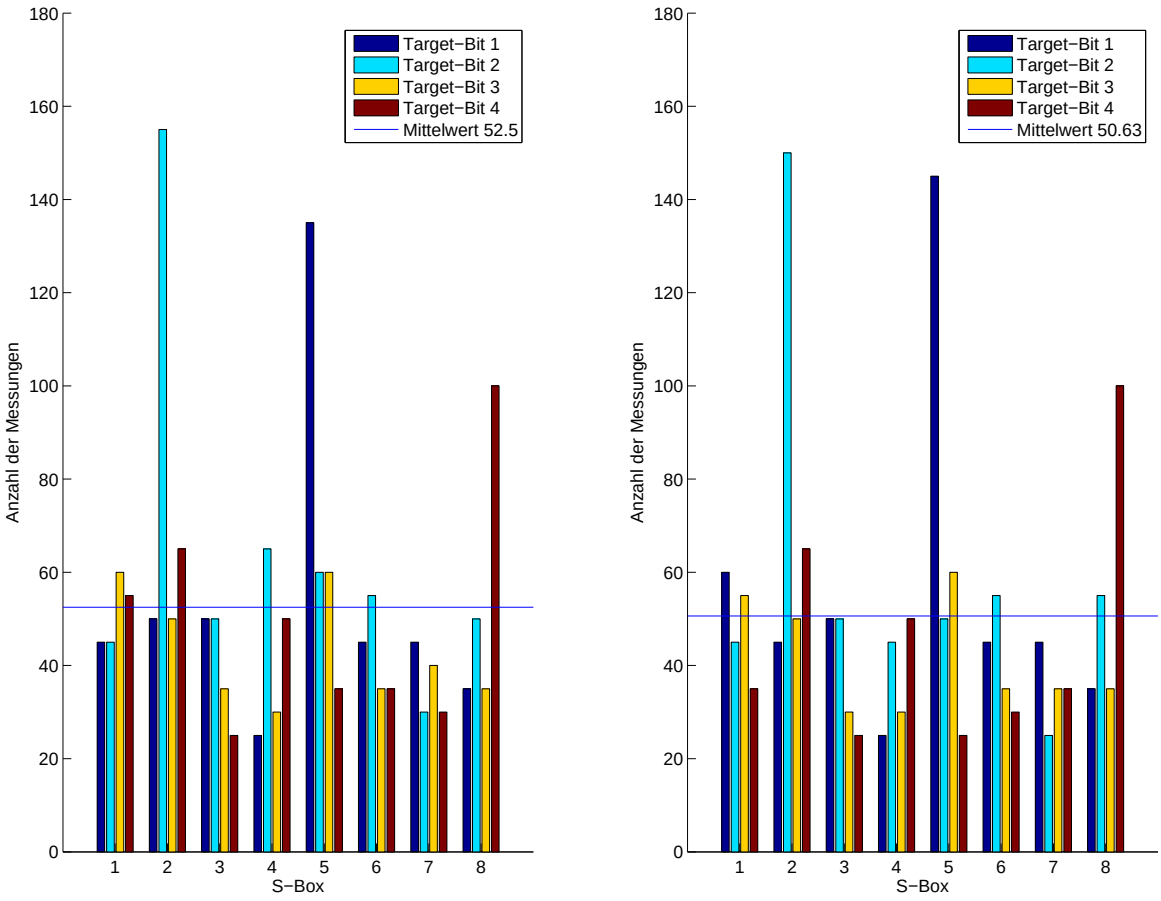

Abbildung 4.10: Anzahl der benötigten Messungen für eine erfolgreiche DPA-Attacke bei einem Downsampling-Faktor von 50 (links) und einem Faktor von 100 (rechts).

Kapitel 5.

ST10F168 – Messaufbau und Ergebnisse

In diesem Kapitel soll der praktische Messaufbau mit dem ST10F168 Mikrocontroller von STMicroelectronics[1] beschrieben werden. Dabei werden die Prozessorarchitektur und die verwendete Schaltungstechnik besprochen.

5.1. Prozessorarchitektur und Entwicklungstools

Beim ST10F168 handelt es sich um einen 16 Bit CISC-Mikrocontroller, der auf der Infineon C16x Architektur basiert. Der Controller verfügt über 256 KB Flash-ROM, 2 KB IRAM sowie über 6 KB XRAM direkt am Chip und kann 16 MB Speicher adressieren. Zusätzlicher Speicher kann über ein flexibles Bus-Interface angeschlossen werden, das einen 8 Bit oder 16 Bit breiten Datenbus nach außen zur Verfügung stellt.

Der Mikrocontroller hat spezielle Peripherieeinheiten eingebaut, die speziell im Automotive-Bereich Anwendung finden:

- Der Interruptcontroller ist sehr ausgereift und bietet eine flexible Konfiguration der Prioritäten der einzelnen Interrupts an. Das ist für Echtzeitanwendungen, wie sie im Auto vorkommen, vorteilhaft.

- Der CAN-Controller verbindet den Mikrocontroller mit einem *Controller Area Network* (CAN). CAN ist ein Bussystem, das für die Kommunikation innerhalb von Automobilen entwickelt wurde. Der Mikrocontroller kann also ohne zusätzliche Hardware mit dem Fahrzeug-Bus kommunizieren.

- Der Controller verfügt über zwei Capture/Compare Units (CAPCOM). Die CAPCOM-Unit ist eine schnelle I/O-Einheit, die I/O-Funktionen parallel zur CPU ausführen kann. Das wird z. B. bei Motorsteuerungen benötigt.

- Der Analog-Digital-Wandler hat eine Auflösung 10 Bit und eine Wandlungszeit von 7,76 μs.

Abbildung 5.1 zeigt das Blockschaltbild des ST10F168, in dem die einzelnen Peripheriekomponenten und die Verbindungen untereinander gezeigt werden. Dieses Blockschaltbild ist dem Datenblatt [STM02] entnommen, in dem eine detaillierte Beschreibung aller Komponenten des Mikrocontrollers zu finden ist.

[1] http://www.st.com

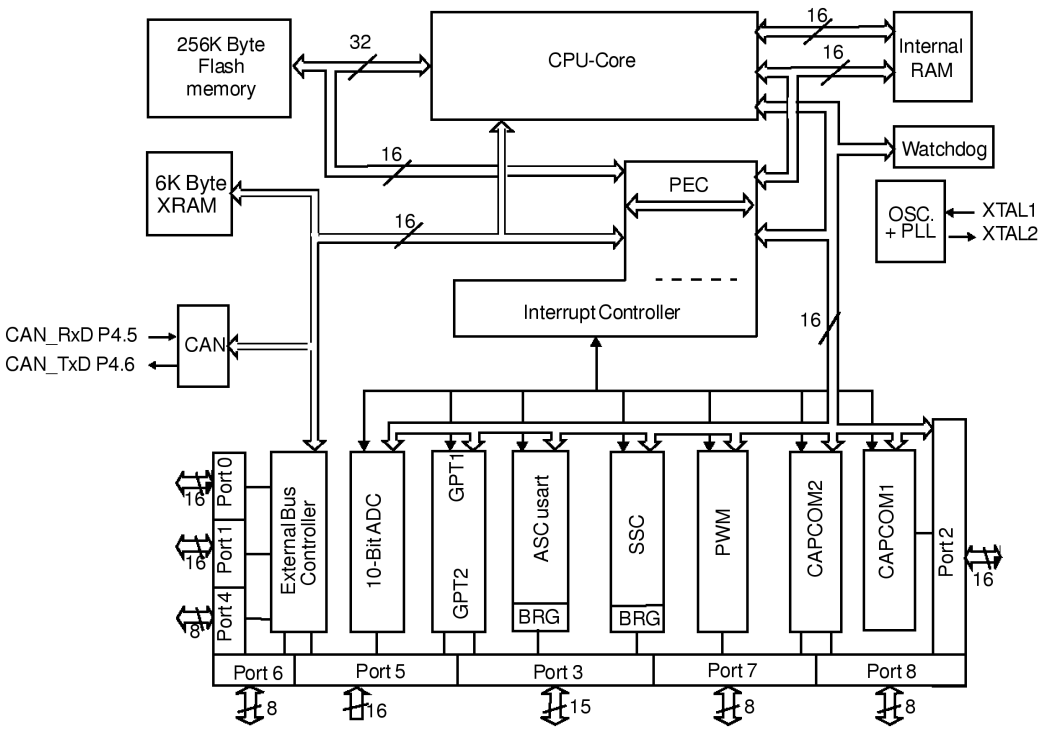

Abbildung 5.1: Blockschaltbild des ST10F168

Entwicklungstools

Als Entwicklungstools kamen die Softwareumgebungen Tasking[2] C166-ST10 v8.0/v8.5 und Keil[3] μVision2/3 mit dem ST10/C16x C-Compiler v5.0 zum Einsatz. Vor Beginn der Diplomarbeit war die Software bereits fertig entwickelt, es war also im Rahmen der Diplomarbeit nicht nötig, sich mit der Softwareentwicklung für den ST10F168 zu beschäftigen.

5.2. Verwendete Hardware

Der Mikrocontroller wird auf dem Evaluationsboard EVA168_2 der Forth-Systeme GmbH[4] betrieben. Das Board enthält neben dem Mikrocontroller unter anderem eine Echtzeituhr, Pegelwandler für die serielle Schnittstelle und die CAN-Schnittstelle, externes Flash-ROM mit einer Größe von 512 KB, externes RAM mit einer Größe von 256 KB sowie Schalter für die Konfiguration des Boards. Der ST10F168 befindet sich in einem Nullkraftsockel, d. h. man kann ihn leicht entnehmen bzw. austauschen. In einem ersten Schritt wurde die Stromaufnahme des kompletten Boards gemessen. Dazu war keine Modifikation des Evaluationsboards nötig. Obwohl mit diesem Setup erfolgreiche DPA-Angriffe möglich waren, empfiehlt es sich, die Strommessung direkt am Prozessor vorzunehmen, da man sonst den Verbrauch der reichlich vorhandenen Peripherie zusätzlich misst. Für die Messung des Stromes direkt am Mikrocontroller ist es nötig, das Board zu modifizieren. Dazu wurde eine Platine angefertigt, die direkt über den Controller mit dem Nullkraftsockel verschraubt wird. Diese Platine dient als Träger für den Shunt-Widerstand und besitzt eine Anschlussmöglichkeit für das Oszilloskop. Der Prozessor besitzt 10 Masseleitungen, die vom Sockel weg gebogen wurden und mit einem feinen Draht mit der Platine verbunden wurden. Nach der Modifikation fließt der Versorgungsstrom über die Platine und somit auch über den Shunt-Widerstand. Abbildung 5.2 zeigt das Evaluationsboard mit aufgeschraubter Zusatzplatine.

Das Board ist mit einem Oszillator mit einer Frequenz von 5 MHz ausgestattet, der den Takt für den Mikrocontroller erzeugt. Der Mikrocontroller verfügt über eine eingebaute PLL-Schaltung zur Frequenzvervielfachung. Die PLL ist von außen über kleine Schalter konfigurierbar und kann auf die Vervielfachungsfaktoren von 1 bis 5 eingestellt werden. Die standardmäßige Einstellung von 5 hat Störungen im Messsignal verursacht, die eine höhere Anzahl von Messkurven für einen erfolgreichen DPA-Angriff erforderten. Deshalb wurde die PLL deaktiviert. Der Mikrocontroller wird also für die Messungen mit einer Taktfrequenz von 5 MHz betrieben, das ergibt eine Periodendauer von 200 ns.

5.3. Analysierte Software

Auf der ST10F168-Plattform wurden drei verschiedene Implementierungen von kryptographischen Algorithmen getestet. Alle drei Implementierungen sind bei Siemens, CT IC 3 entstanden. Es handelt sich dabei um eine DES-Implementierung ohne Seitenkanalresistenz (siehe [MvOV96] und [NIS99]) und eine seitenkanalresistente DES-Implementierung mit einer Mas-

[2] http://www.tasking.com
[3] http://www.keil.com
[4] http://www.fsforth.de

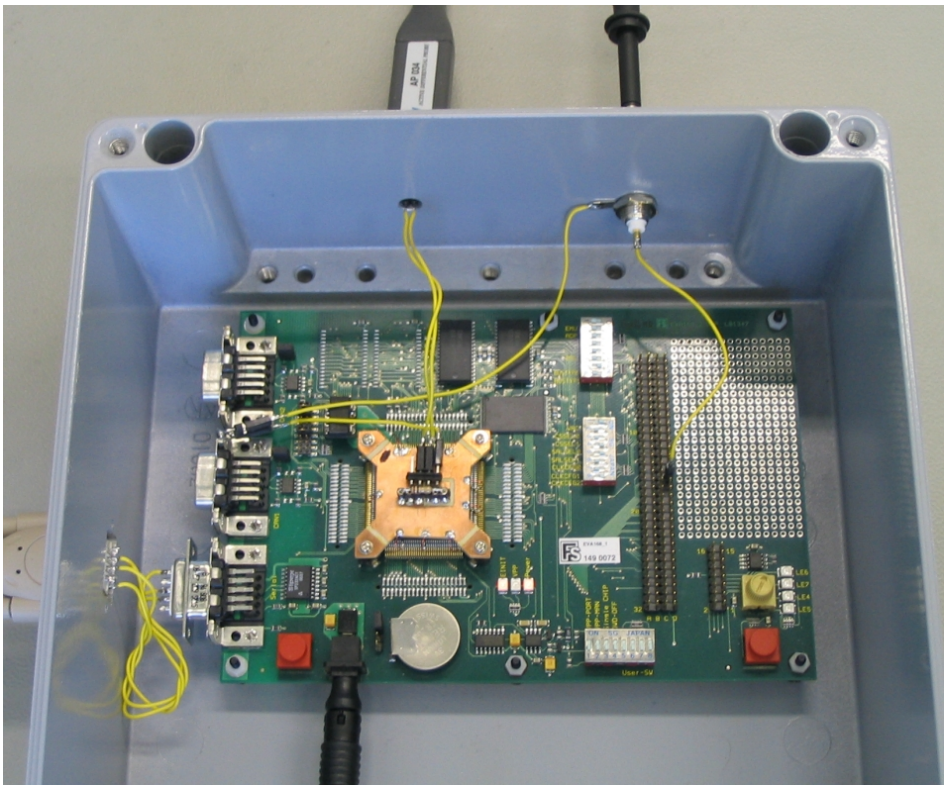

Abbildung 5.2: Foto des modifizierten Evaluationsboards

kierung analog zu [AG01]. Beim dritten Algorithmus handelt es sich um eine nicht seitenkanalresistente 16 Bit Implementierung von AES (siehe [NIS01] und [DR99]).

5.4. Messergebnisse

5.4.1. Charakterisierung der Stromaufnahme

Abbildung 5.3 zeigt die Stromaufnahme während der Berechnung einer DES-Operation. Die Daten stammen direkt vom Oszilloskop und wurden mit einer Abtastrate von 200 $^{\mathrm{MS}}/_{\mathrm{s}}$ aufgezeichnet. Die Daten wurden nicht nachbearbeitet. Die 16 DES-Runden sind deutlich erkennbar.

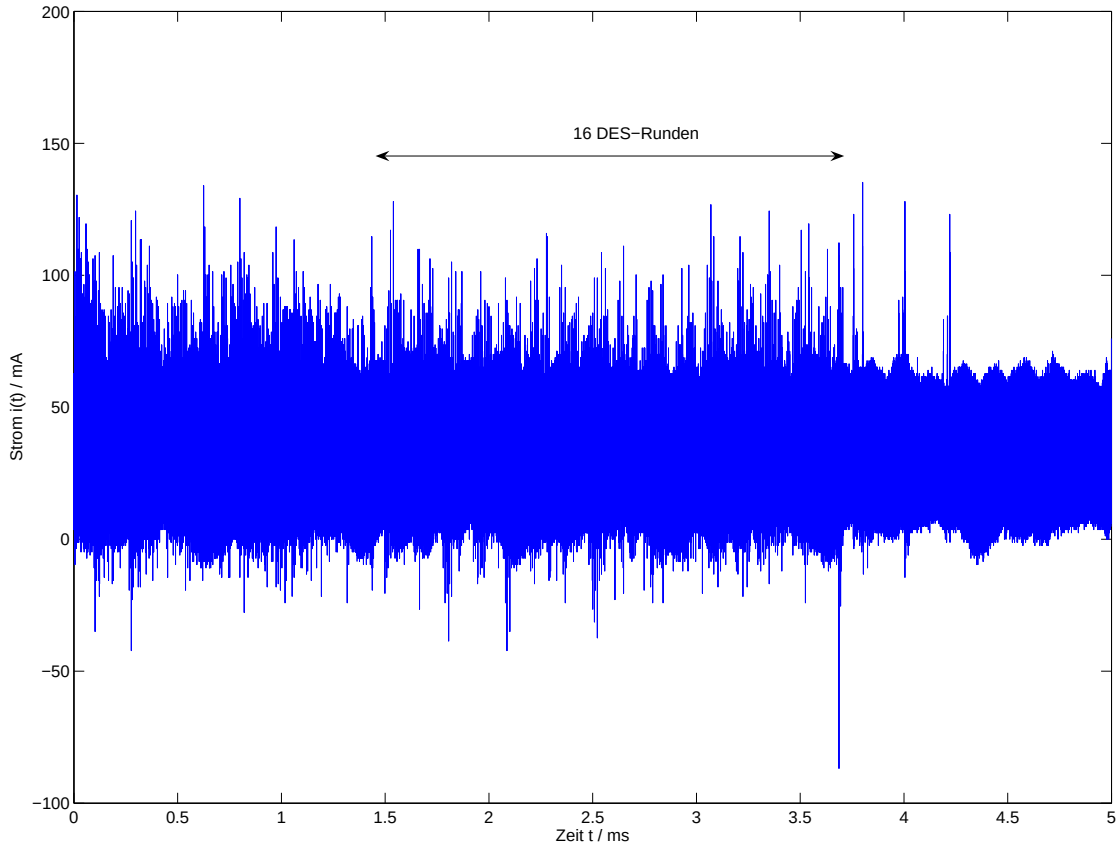

Abbildung 5.3: Die Stromaufnahme des ST10F168 auf einem Evaluationsboard

Das Spektrum der Stromaufnahme ist in Abbildung 5.4 dargestellt. Die Amplitudenwerte sind nicht auf eine physikalische Größe normiert und werden linear dargestellt, da sich bei dieser Darstellungsform die Taktfrequenz und die Oberwellen am deutlichsten zeigen. Man sieht, dass die größten Frequenzanteile im unteren Frequenzbereich angesiedelt sind und die Signalbandbreite schmal ist. Auffallend ist die Spitze bei der halben Taktfrequenz von 2,5 MHz und die sehr ausgeprägte Spitze bei 10 MHz. Dieses Spektrum wurde aus einer Stromkurve entnommen, die bei der Verwendung eines Gehäuses gemessen wurde. Das Spektrum ohne Gehäuse unterscheidet sich nur minimal und wird an dieser Stelle nicht gezeigt.

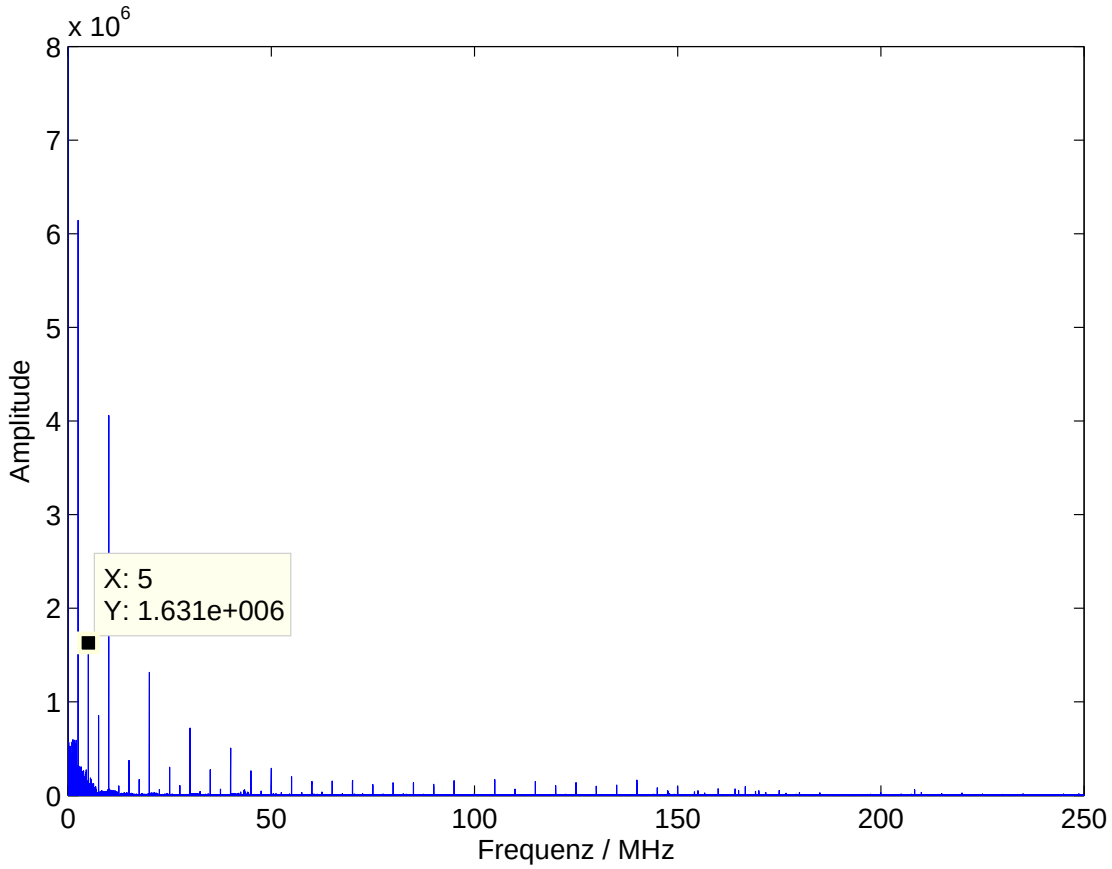

Abbildung 5.4: Das Spektrum der Stromaufnahme des ST10F168 auf einem Evaluationsboard

5.4.2. Angriff auf eine exemplarische S-Box

In diesem Abschnitt soll exemplarisch für eine S-Box beim DES und für ein Byte beim AES ein DPA-Angriff mit der Korrelationsmethode durchgeführt werden. Der Angriff auf den DES erfolgte auf die erste S-Box und das erste Target-Bit. Der zugehörige richtige Teilschlüssel hat den Wert 38. Abbildung 5.5 und Abbildung 5.6 zeigen jeweils die Ergebnisse von Attacken auf den DES-Algorithmus, bei dem im ersten Fall die Maskierung deaktiviert und im zweiten Fall aktiviert war. Im oberen Teil der Abbildungen sind 200 Messkurven ($N = 200$) in die Analyse eingeflossen und im unteren Teil waren es 1000 Messkurven ($N = 1000$). Die Messkurven wurden mit einer Abtastfrequenz von 1 $^{\mathrm{GS}}/_{\mathrm{s}}$ (= 200-faches Oversampling) aufgezeichnet, damit das Rauschen, verursacht durch Phasenjitter, klein gehalten wird. Anschließend wurden die Daten einem Downsampling mit dem Faktor 50 unterzogen. Auf der linken Seite der Abbildungen wird jeweils die Korrelationskurve der wahrscheinlichsten Schlüsselhypothese gezeigt. In Abbildung 5.5 wird dabei in beiden Fällen der richtige Teilschlüssel gefunden. Bei der maskierten Version in Abbildung 5.6 zeigt sich dagegen keine Spitze und der richtige Teilschlüssel konnte nicht gefunden werden. Durch das Hinzufügen von Messungen verringert sich lediglich das Rauschen. Der Schlüssel kann mit einer einfachen DPA-Attacke (First Order) jedoch nicht gefunden werden. Auf der rechten Seite der beiden Abbildungen sind die Absolutwerte der Spitzenwerte aller 64 Korrelationskurven dargestellt[5].

In Abbildung 5.7 ist das Ergebnis einer DPA-Attacke mit Korrelationsmethode auf den AES-Algorithmus dargestellt. Auf der linken Seite sind wieder die Korrelationskurven für 200 und 1000 Messungen abgebildet. Die Daten wurden mit einer Samplingfrequenz von 2 $^{\mathrm{GS}}/_{\mathrm{s}}$ aufgezeichnet und anschließend einem Downsampling mit dem Faktor 40 unterzogen. Rechts sind die Absolutwerte der Maxima aller 256 Korrelationskurven dargestellt.

5.4.3. Anzahl der benötigten Messungen

Die Anzahl der benötigten Messungen bis das Ergebnis der DPA-Analyse konvergiert, hängt nicht nur von der Qualität der Messungen ab, sondern auch von der angegriffenen S-Box und dem angegriffenen Bit. In diesem Abschnitt wird gezeigt, wie viele Messungen auf dem Evaluationsboard des ST10F168 nötig sind, damit ein Angreifer einen entsprechenden Teilschlüssel finden kann.

In Abbildung 5.8 wird die Anzahl der benötigten Messungen für einen erfolgreichen Angriff auf den DES-Algorithmus dargestellt. Dabei wird ein Messdatensatz, der ohne die Verwendung eines Abschirmgehäuses entstanden ist, einem Messdatensatz, der mit einem Abschirmgehäuse entstanden ist, gegenübergestellt. Das Ergebnis zeigt, dass bei Verwendung eines Gehäuses minimal mehr Messungen nötig sind. Das ist auf die längere Messleitung vom Tastkopf zum Shunt-Widerstand zurückzuführen. Die lange Messleitung macht den Gewinn durch das Gehäuse mit der vergrößerten Kopplung innerhalb des Gehäuses zunichte.

Abbildung 5.9 zeigt die Anzahl der benötigten Messungen, um einen erfolgreichen Angriff auf den AES-Algorithmus durchführen zu können. Als Schlüssel kam der folgende Wert zum Einsatz `0x11223344556677889aabbccddeeff25`.

Für die Ermittlung dieses Diagramms wurden 1000 Messungen ausgewertet. Bei manchen Byte/Bit-Kombinationen reichen auch 1000 Messungen nicht aus, damit ein erfolgreicher An-

[5] Bei der gesicherten Implementierung war selbst eine „Higher Order DPA“ wie in [Mes00b] mit genauer Kenntnis der Implementierung nicht erfolgreich. Bei diesem Angriff wurden 10000 Messungen durchgeführt.

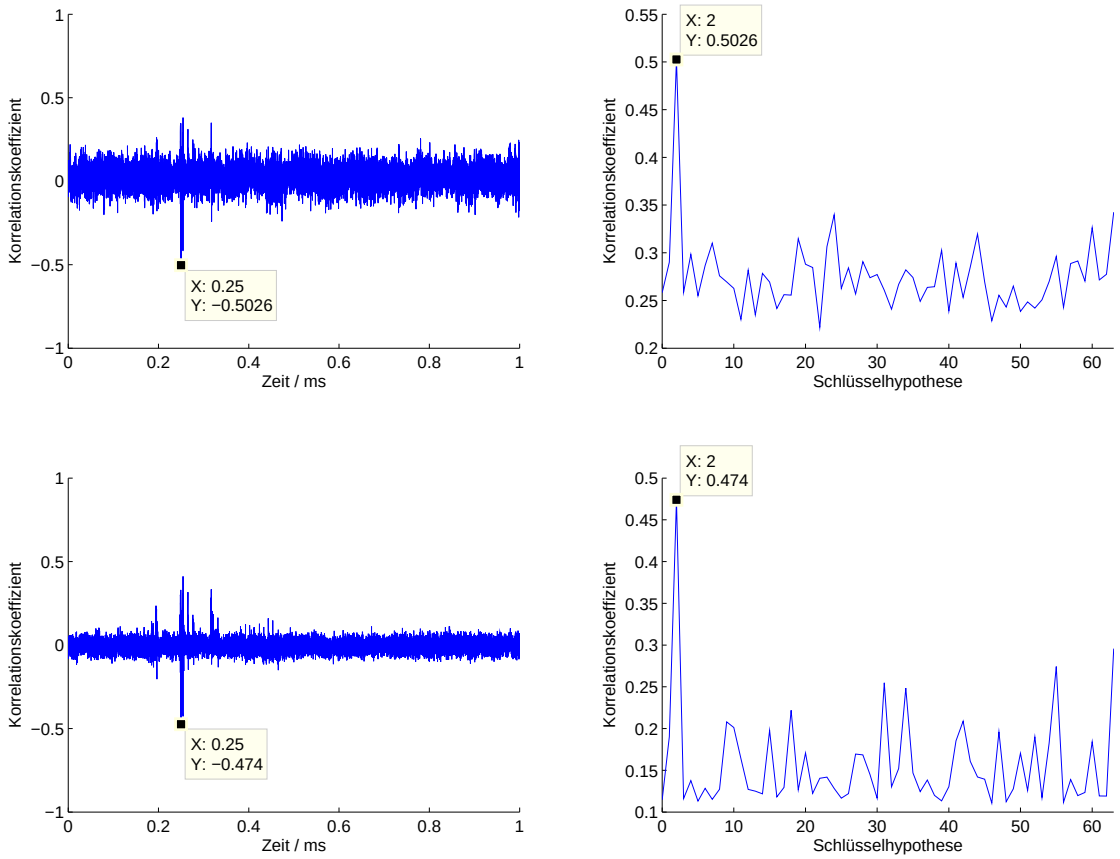

Abbildung 5.5: Ergebnis einer DPA-Analyse bei deaktivierter Maskierung der Daten. Im oberen Teil der Abbildung sind 200 Messkurven in die Analyse eingeflossen. Im unteren Teil wurden 1000 Messkurven analysiert. Links sind die wahrscheinlichsten Korrelationskurven und rechts die Absolutwerte der Maxima aller 64 Korrelationskurven dargestellt.

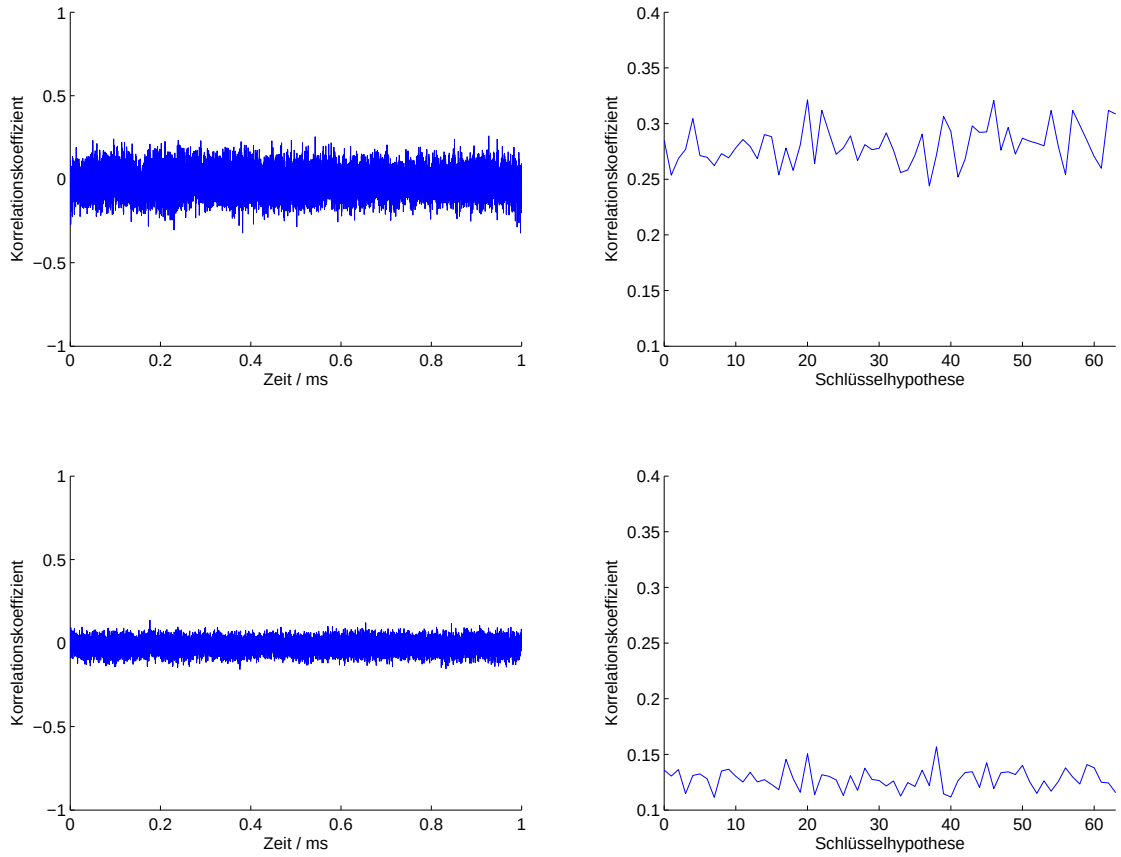

Abbildung 5.6: Ergebnis einer DPA-Analyse bei aktivierter Maskierung der Daten. Im oberen Teil der Abbildung sind 200 Messkurven in die Analyse eingeflossen. Im unteren Teil wurden 1000 Messkurven analysiert. Links sind die wahrscheinlichsten Korrelationskurven und rechts die Absolutwerte der Maxima aller 64 Korrelationskurven dargestellt.

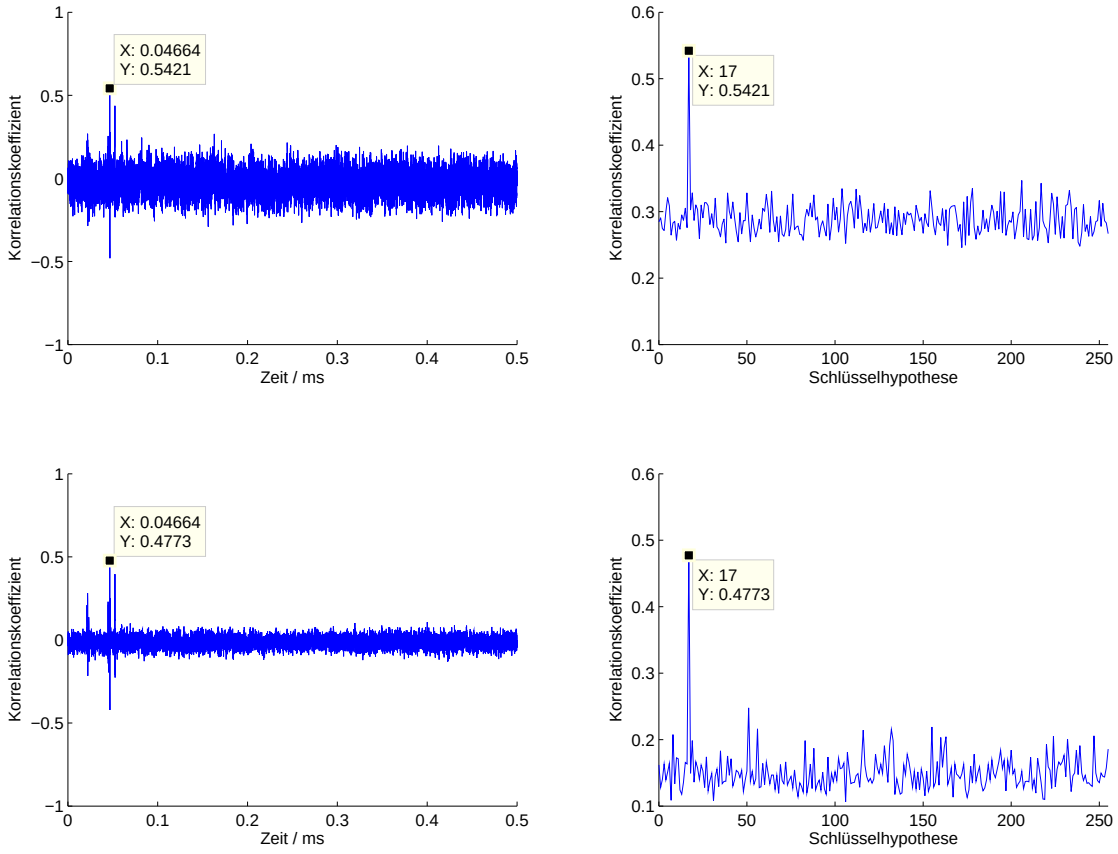

Abbildung 5.7: Ergebnis einer DPA-Analyse einer nicht seitenkanalresistenten AES-Implementierung. Links sind die Korrelationskurven für 200 und 1000 Messungen dargestellt. Rechts sieht man die Absolutwerte der Spitzenwerte aller 256 Korrelationskurven.

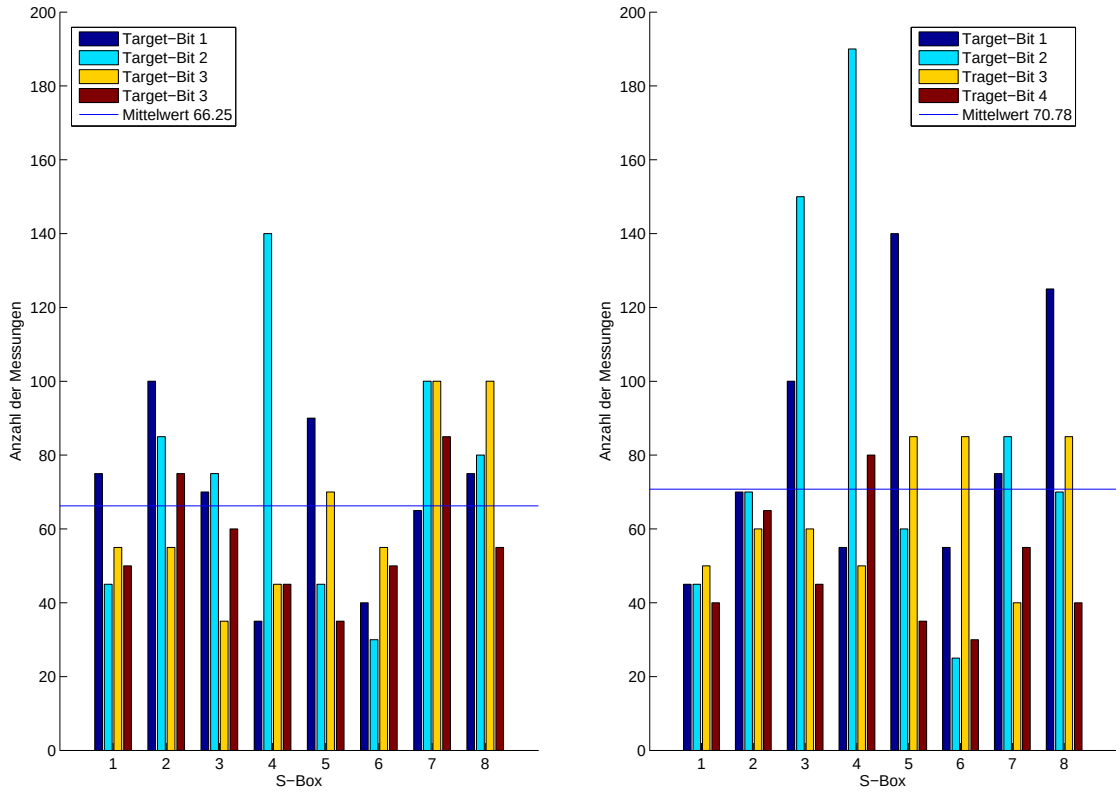

Abbildung 5.8: Anzahl der benötigten Messungen beim DES abhängig von der angegriffenen S-Box und einem der vier Target-Bits. Für das Ergebnis auf der linken Seite wurden Messungen ohne Gehäuse durchgeführt, rechts wurde mit Gehäuse gemessen.

griff durchgeführt werden kann. Im Diagramm ist das an den Balken, die bis zum Wert 1000 reichen, zu erkennen. Es ist klar ersichtlich, dass man für einen erfolgreichen Angriff auf die verwendete AES-Implementierung mehr Messungen benötigt als für einem Angriff auf den DES. Uns sind jedoch Angriffe auf AES-Implementierungen bekannt, bei denen man mit deutlich weniger Messungen auskommt. Das hier gewonnene Ergebnis liegt wahrscheinlich an der speziellen speichersparenden Implementierung des AES. Bei dieser Variante werden Operationen, die man mit Tablelookups ausführen könnte, tatsächlich als Berechnungen vom Mikrocontroller ausgeführt.

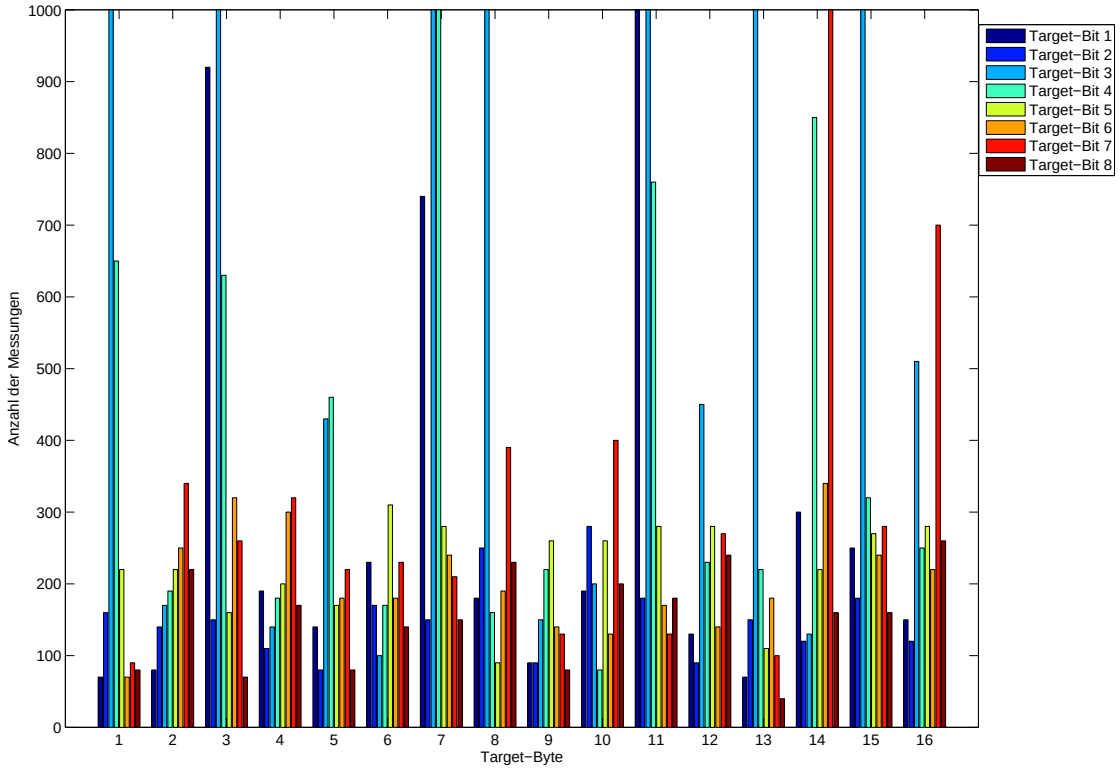

Abbildung 5.9: Anzahl der benötigten Messungen beim AES abhängig vom angegriffenen Byte und Target-Bit. Beim AES-Algorithmus gibt es jeweils acht Target-Bits bei 16 Bytes, die angegriffen werden können.

Kapitel 6.

Chipkarten – Messaufbau und Ergebnisse

In diesem Kapitel soll der praktische Messaufbau für das Analysieren von Chipkarten vorgestellt werden. Die Grundidee ist, für die Kommunikation zwischen PC und Chipkarte ein handelsübliches Lesegerät zu verwenden, das wegen der Verwendung eines Adapters nicht modifiziert werden muss. Das Chipkartenprotokoll wird von einem Mikrocontroller analysiert, der zu gegebenem Zeitpunkt das Oszilloskop triggert. Die Ansteuerung des Chipkartenlesegerätes erfolgt über die PC/SC Softwareschnittstelle der `@target_comm` Klasse von Matlab aus. Abbildung 6.1 soll dieses Konzept visuell darstellen.

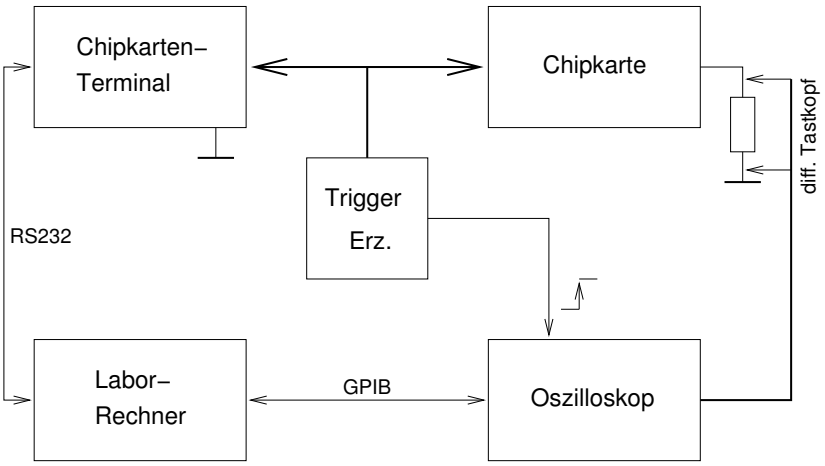

Abbildung 6.1: Grundkonzept des Chipkartenmessaufbaus

6.1. Chipkartenprotokolle

Die Kommunikation zwischen Chipkarte und Terminal erfolgt seriell über eine einzige Datenleitung. Somit können die zwei Kommunikationspartner nur wechselseitig kommunizieren. Der typische Kommunikationsablauf ist in Abbildung 6.2 dargestellt. Nach dem Reset sendet die Chipkarte die *ATR* (Answer to Reset) an das Terminal zurück, in dem die Karte dem Terminal ihre gewünschten Kommunikationsparameter mitteilt. Als zweiten Schritt können optional mittels *PPS* (Protocoll Parameter Selection) die Kommunikationsparameter vom Terminal geändert werden. Mit einer Antwort bestätigt oder verweigert die Karte die gewünschten Einstellungen. Wenn die Parameter erfolgreich ausgehandelt wurden, kann die eigentliche Kommunikation starten, die nach dem Schema „Kommando-Antwort“ abläuft. Wie zu sehen

ist, wird jeder Schritt vom Terminal initiiert. Alle in diesem Abschnitt geschilderten Abläufe entstammen dem Handbuch der Chipkarten [RE02].

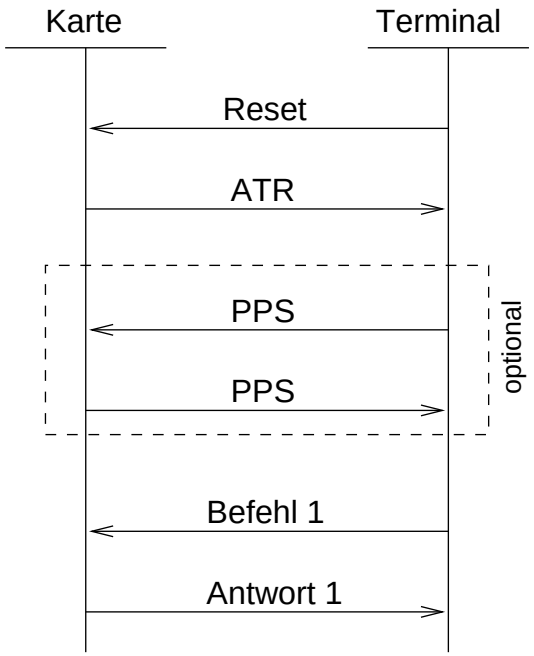

Abbildung 6.2: Kommunikationsablauf zwischen Karte und Terminal

6.1.1. Physikalische Übertragungsschicht (OSI Layer 1)

Die physikalische Übertragungsschicht ist in der Chipkartennorm ISO 7816-3 festgelegt. Die Kommunikation erfolgt mittels eines seriellen digitalen Protokolls, bei dem die Bits mit den Spannungspegeln 0 V und Versorgungsspannung dargestellt werden. Die Versorgungsspannung kann +5 V, +3 V oder +1,8 V betragen. Die Zuordnung, ob 0 V oder die Versorgungsspannung eine logische 0 oder 1 darstellt, ist variabel und wird im ATR festgelegt. Auch die Bitrate, mit der die Kommunikation erfolgt, wird im ATR festgelegt. Beim Senden des ATR ist sie jedoch auf 9600 Baud festgelegt. Die Datenübertragung erfolgt asynchron, das bedeutet, dass zu jedem Datenwort Synchronisationszeichen hinzugefügt werden müssen. Ein Datenwort wird mit einem Startbit eingeleitet, darauf folgen acht Datenbits plus einem Paritätsbit. Abgeschlossen wird das Zeichen mit einem oder zwei Stopbits. Dieser Kommunikationsablauf ist identisch mit dem einer RS232 Schnittstelle, nur die Parameter sind bei RS232 Schnittstellen unüblich. Es ist jedoch möglich, einen *UART* (Universal Asynchronous Receiver Transmitter) so zu konfigurieren, dass die serielle Kommunikation abgehört werden kann. Genau dies wurde im Rahmen dieser Arbeit auch getan, die UART-Peripherieeinheit eines ATmega16-Mikrocontrollers wurde so programmiert, dass der Mikrocontroller die Kommunikation abhören, analysieren und im richtigen Moment ein Triggersignal auslösen konnte.

6.1.2. ATR und PPS

Wie schon in der Einleitung erwähnt, dienen ATR und PPS dem Aushandeln der Kommunikationsparameter zwischen Karte und Terminal. Eine genaue Beschreibung von ATR und PPS würde den Rahmen dieser Arbeit sprengen. Für genaue Ausführungen sei auf das Buch [RE02] verwiesen. Hier soll nur die Bedeutung und der Aufbau dieser beiden Datenblöcke so weit erläutert werden, dass es für das weitere Verständnis ausreichend ist. Der ATR-Block wird jedes Mal nach dem Reset von der Karte zum Terminal gesendet. Die Länge ist dabei dem Terminal nicht bekannt und ist erst aus dem aktuellen ATR erkenntlich. Jedoch wird die Länge nicht in einem Header codiert, sondern es gibt zwischendurch immer Datenbits, die festlegen, welche Informationen als nächstes folgen. Der ganze Ablauf ist sehr kompliziert und erfordert das Einlesen des gesamten ATR-Blocks, damit man die Länge feststellen kann. Für die Analyse der Datenkommunikation ist es jedoch nötig, die Länge des ATR-Blocks zu kennen, damit man den Anfang des ersten Datenblocks finden kann. Der PPS-Block hat einen ähnlichen Aufbau wie der ATR Block, jedoch ist der PPS kürzer und kann eine Länge von drei bis sechs Bytes aufweisen.

6.1.3. Das Übertragungsprotokoll T=1 (OSI Layer 2)

Die gängigsten Chipkartenprotokolle bei Prozessorkarten sind T=0 und T=1. T=0 ist ein byteorientiertes Protokoll und ist in ISO 7816-3 spezifiziert. Es wurde in Frankreich entwickelt und kommt z. B. bei GSM SIM-Karten zum Einsatz. T=1 ist ebenfalls in ISO 7816-3 spezifiziert und wurde in Deutschland entwickelt, es ist im Gegensatz zu T=0 blockorientiert. In dieser Arbeit wurde ausschließlich das Protokoll T=1 verwendet, deshalb beschränken sich die Ausführungen in diesem Abschnitt auf T=1.

Von einem Datenblock, wie er in Abbildung 6.3 zu sehen ist, gibt es drei Ausprägungen. Der Informationsblock (I-Block) dient dem transparenten Austausch von Daten der Anwendungsschicht (OSI Layer 7) in Form von APDUs. Die Abkürzung APDU steht für den englischsprachigen Ausdruck „application protocol data unit“ und bezeichnet einen Datenblock der Anwendungsschicht (OSI Layer 7). Der Empfangsbestätigungsblock (R-Block) besitzt kein Informationsfeld und wird für die positive und negative Empfangsbestätigung verwendet. Die dritte Ausprägung ist der so genannte Systemblock (S-Block) und wird für die Übermittlung von Steuerinformationen, die das Protokoll selbst betreffen, herangezogen. Um welche Art von Block es sich handelt, wird im Protokollkontrollbyte (PCB) festgelegt. Die Knotenadresse (NAD) enthält die Quell- und Zielknotenadresse und wird in der Regel nicht mehr verwendet und auf 0x00 gesetzt. Das LEN Byte gibt die Länge des folgenden Datenblocks (APDU) an, das Epilogfeld enthält eine Checksumme zur Fehlererkennung.

Prologfeld			Informationsfeld	Epilogfeld
Knotenadresse NAD	Protokollkontrollbyte PCB	Länge LEN	APDU	Checksumme EDC
1 Byte	1 Byte	1 Byte	0 ... 254 Byte	1 ... 2 Byte

Abbildung 6.3: Der Aufbau eines T=1 Übertragungsblocks

6.1.4. Struktur der Kommando-APDUs (OSI Layer 7)

APDUs werden als Nutzinformation eines darunter liegenden Protokolls übertragen. Im Falle von T=1 werden APDUs im Informationsfeld eines I-Blocks übertragen. Eine APDU enthält ein Kommando der Anwendungsschicht und kann je nach Art des Kommandos zusätzlich ein Datenfeld enthalten, in dem Parameter oder Nutzdaten übertragen werden können. Abbildung 6.4 zeigt den Aufbau einer Kommando-APDU (a) und die vier möglichen Ausprägungen (b1–b4), die vom übertragenen Befehl abhängig sind.

Im Header, der auf jeden Fall übertragen wird, ist der zu übertragende Befehl codiert. CLA wählt die Befehlsklasse aus. Im Standard sind Klassen für verschiedene Anwendungen festgelegt. Die Klasse mit der Nummer 0x80 ist für anwenderspezifische Befehle reserviert. Das Byte INS kodiert den eigentlichen Befehl, es sind also 256 Befehle pro Klasse möglich. In den Feldern P1 und P2 werden zusätzliche Parameter für den Befehl übertragen. Der Body besteht aus drei Feldern und wird je nach Befehl voll, teilweise oder nicht übertragen (siehe Abbildung 6.4 b1–b4). Das Lc-Feld (length command) gibt an, wie groß das darauffolgende Datenfeld sein wird, als logische Folge treten die beiden Felder immer gemeinsam auf (Abbildung 6.4 (b3), (b4)). Im Le-Feld (length expected) wird die Länge einer zu erwartenden Antwort auf den Befehl angegeben. Wenn auf den Befehl keine Antwort erwartet wird, entfällt das Le-Feld.

Die Software zur Erzeugung des Triggersignals beschränkt sich auf das Erkennen einer solchen APDU. Nach vollständigem Senden der APDU, also in dem Moment, da die Chipkarte das Kommando zur Verschlüsselung eines Datenworts erhalten hat, wird das Triggersignal erzeugt.

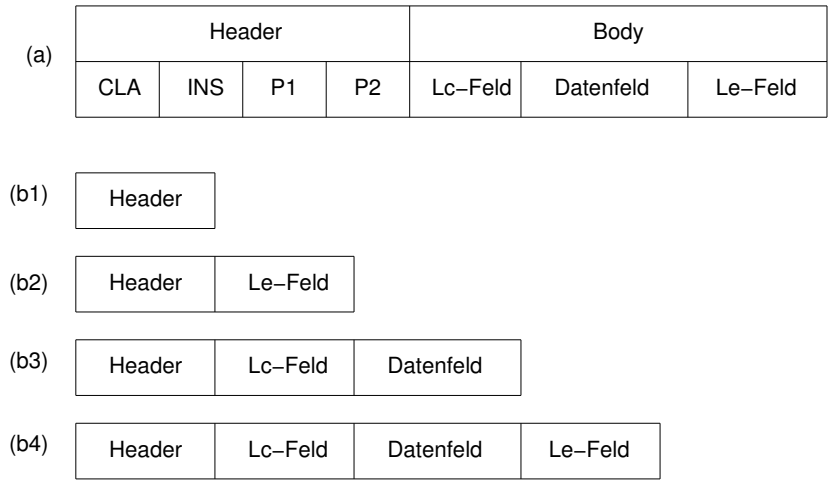

Abbildung 6.4: Der Aufbau einer Kommando-APDU (a) und verschiedene mögliche Ausprägungen (b1–b4)

6.2. Triggererzeugung

Nachdem der grundlegende Ablauf einer Chipkartenkommunikation erläutert wurde, soll nun gezeigt werden, wie man mittels einer Hardware/Software-Lösung den richtigen Triggerzeitpunkt für eine DPA-Messung ermitteln kann. Bei dieser Lösung wird davon ausgegangen, dass

die Berechnung zwischen dem Eintreffen des Kommandos bei der Chipkarte und dem Senden des Ergebnisses an das Terminal erfolgen muss. Das Triggersignal ist während dieser beiden Ereignisse auf logisch 1 gesetzt, was eine Messung der Dauer zwischen der positiven und der negativen Flanke des Signals ermöglicht. Das Oszilloskop wird mit der positiven Flanke zu Beginn der Berechnung gestartet.

Die hier vorgestellte Methode zur Überwachung der Chipkartenkommunikation ist auf eine Übertragungsgeschwindigkeit von 9600 Baud beschränkt. Durch eine Erweiterung der Software wäre es auch möglich, die Chipkartenkommunikation bei anderen Übertragungsraten zu analysieren.

6.2.1. Hardware

Abbildung 6.5 zeigt ein Blockschaltbild, welches die Signalflüsse zwischen Terminal, Chipkarte und Triggerschaltung darstellt. Wie zu sehen ist, werden die Datenleitung und die Resetleitung von der Schaltung überwacht. Beide Leitungen sind mit externen Interrupt-Pins des Mikrocontrollers verbunden. Die Datenleitung ist zusätzlich mit dem Empfangspin des UART verbunden. Interrupt 0 (INT0) ist im Grundzustand nicht aktiviert und wird erst aktiv, wenn es darum geht, die Triggerleitung wieder auf logisch 0 zu setzen. Interrupt 2 (INT2) ist immer aktiv und bringt die Software nach einem Reset der Chipkarte wieder in den Ausgangszustand zurück. Bei Empfang eines Datenbytes wird ebenfalls ein Interrupt ausgelöst, in dessen Interruptserviceroutine (ISR) das Byte analysiert wird. Die zugehörige Umsetzung als Schaltung ist in Abbildung 6.6 zu sehen.

Für den Messaufbau wurde keine besondere Abschirmung oder galvanische Trennung von Schaltungsteilen vorgenommen. Lediglich beim Aufbau der Schaltung wurde darauf geachtet, Kopplungen zwischen Leitungen nach Möglichkeit zu verhindern. Die Messung der Stromaufnahme erfolgt sehr nahe an der Chipkarte. Die Erzeugung des Triggersignals erfolgt räumlich getrennt auf einer separaten Platine, die zugleich als Adapter für das Chipkartenterminal verwendet wird. Abbildung 6.7 zeigt den Messaufbau bei einer Messung an einer Chipkarte.

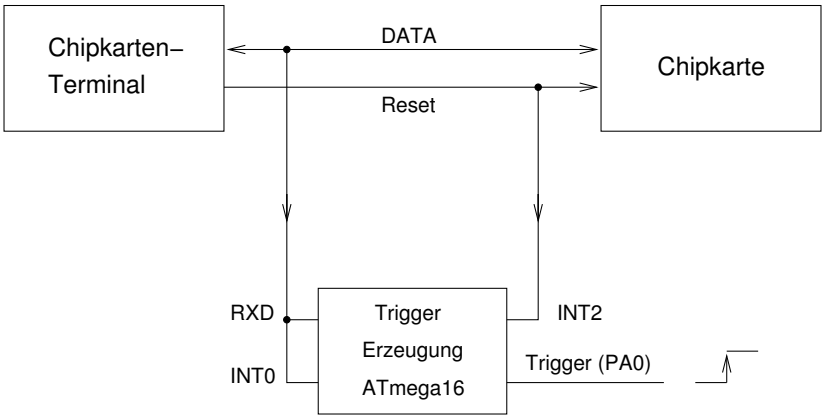

Abbildung 6.5: Blockschaltbild der Triggererzeugung

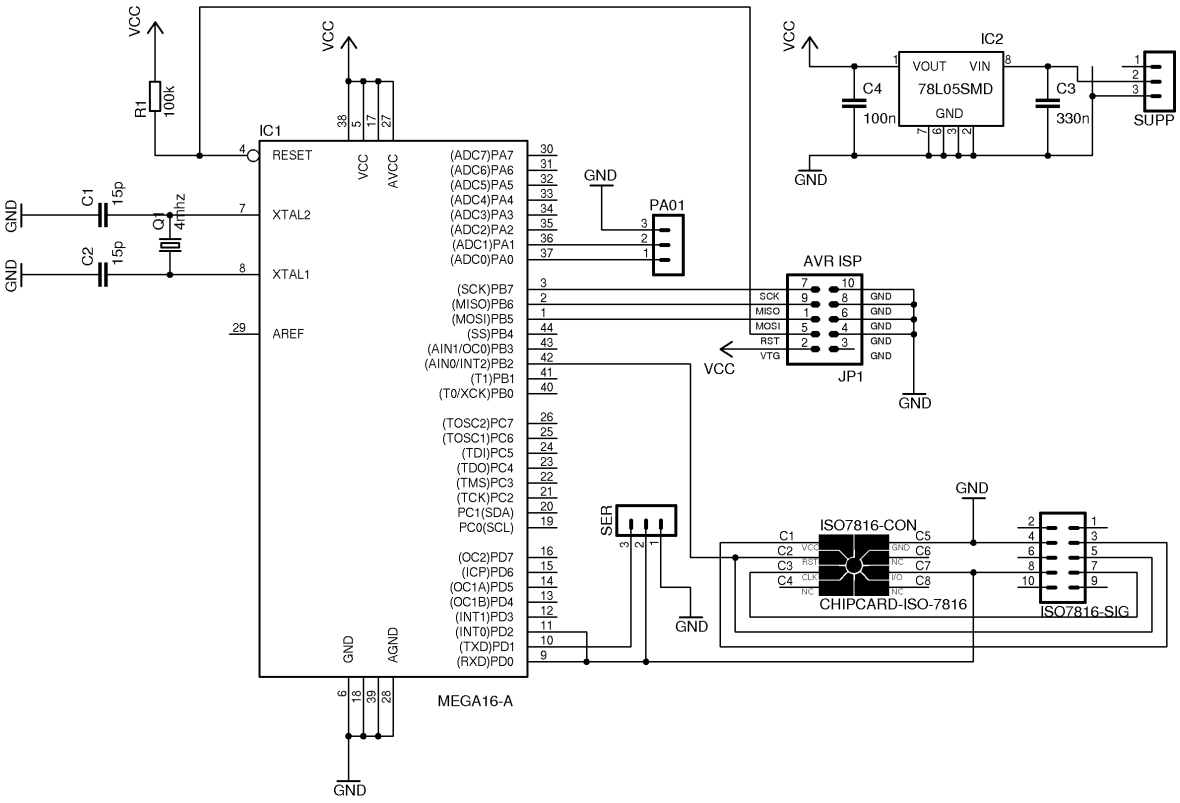

Abbildung 6.6: Schaltplan der Triggererzeugung

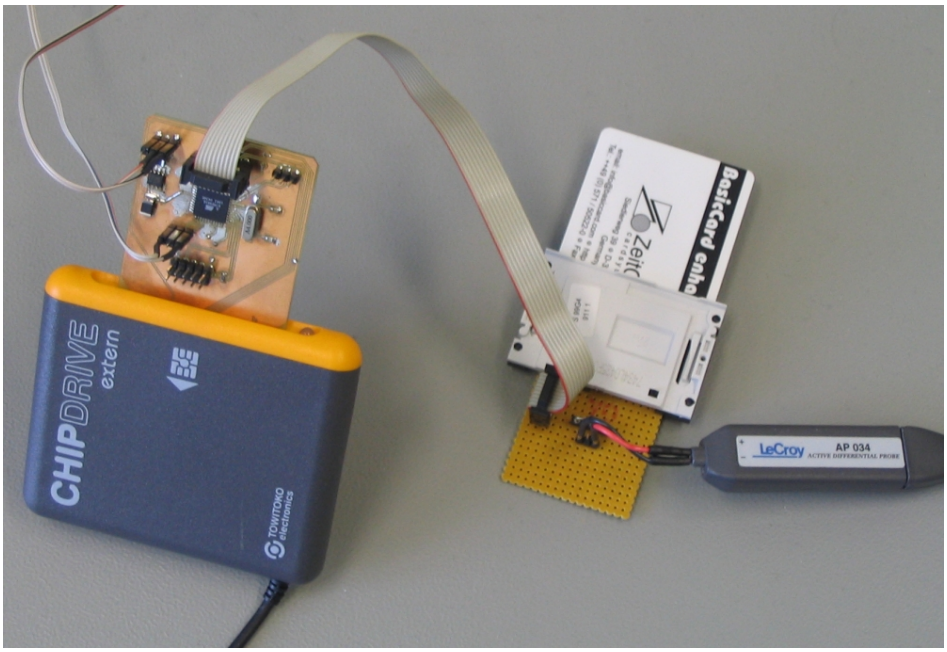

Abbildung 6.7: Foto des Chipkarten-Messsetups

6.2.2. Software

Wie schon im vorigen Abschnitt kurz erwähnt, reagiert die Software auf drei externe Interruptquellen. Der ganze Ablauf der Software wird durch Interruptserviceroutinen gesteuert. Die Analyse des Protokolls erfolgt über zwei Zustandsautomaten, bei denen die drei Interrupts Zustandsübergänge auslösen. Abbildung 6.8 zeigt die zwei Zustandsautomaten als Zustands-Übergangs-Diagramme. Links ist der globale Automat zu sehen, der die gesamte Kommunikation überwacht und im richtigen Moment das Triggersignal aktiviert. Rechts ist der Automat für die Überwachung der ATR-Sequenz zu sehen. Der rechte Automat läuft ab, wenn der linke im Zustand „ATR“ ist. Dieser hierarchische Aufbau wurde aus Gründen der Übersichtlichkeit gewählt. Fast jeder Übergang wird durch einen Empfangsinterrupt der seriellen Schnittstelle ausgelöst, das empfangene Byte entscheidet über den weiteren Ablauf des Automaten.

Wenn bei einem Übergang die Variable `c` angegeben wird, so bedeutet das, dass mehrere Bytes in diesem Zustand empfangen werden müssen, damit der Automat in den nächsten Zustand wechseln darf. Die Variable `c` ist von manchen empfangenen Bytes abhängig (z. B. einem LEN-Byte) und wird bei jedem empfangenen Byte um eins dekrementiert. Wenn der Zähler den Wert 0 erreicht, darf in den nächsten Zustand gewechselt werden.

Im Zustand „BODY“ wird die APDU in einen Buffer geschrieben und kann zur Analyse der Kommunikation an den PC geschickt werden, oder es kann der Trigger abhängig von einer bestimmten APDU gesetzt werden.

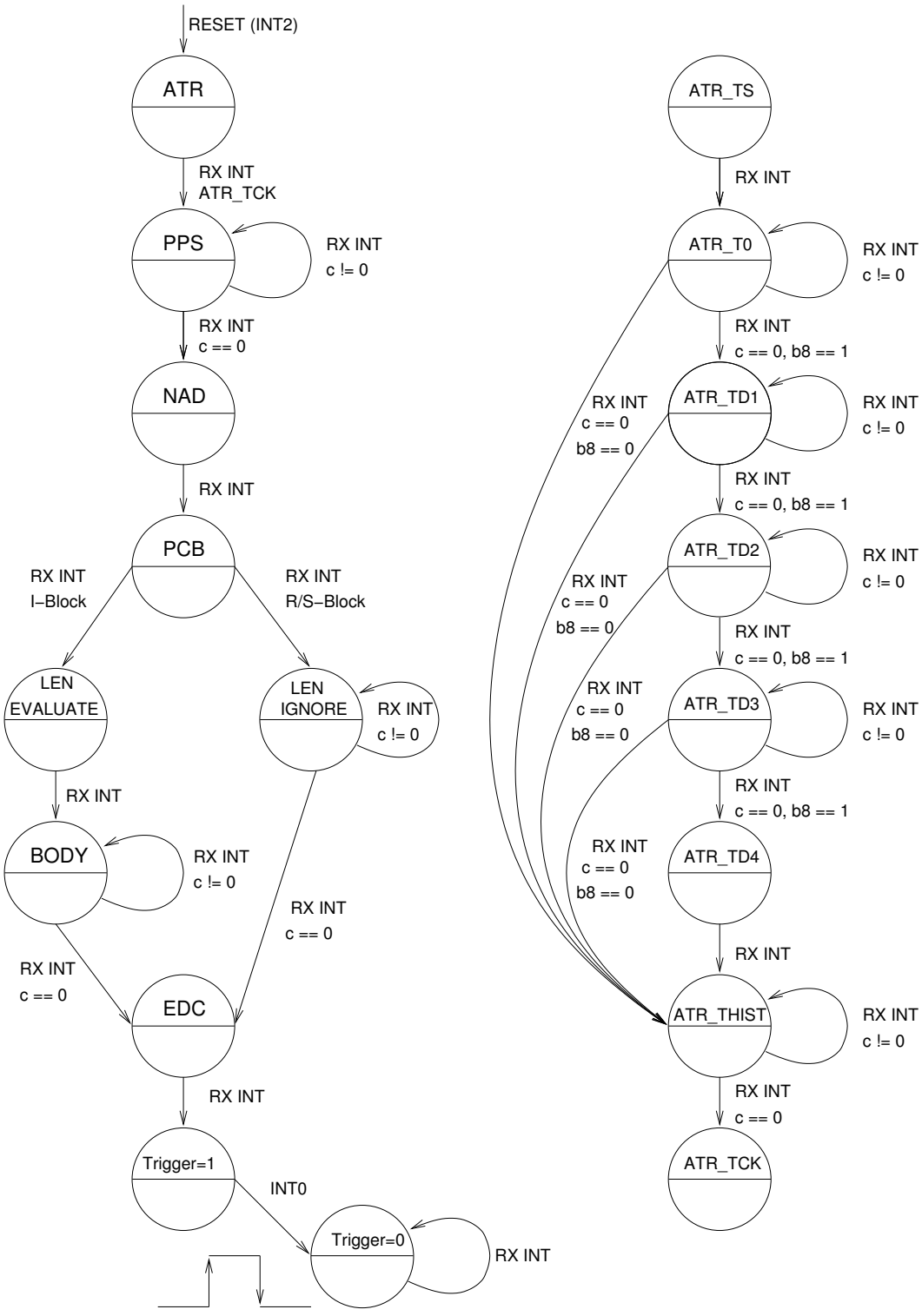

Abbildung 6.8: Zustandsautomat der Software zur Triggererzeugung bei Messungen an den Chipkarten

6.2.3. Messung der Datenleitung und des Triggersignals

In diesem Abschnitt wird durch eine Messung gezeigt, wie das Ergebnis der zuvor beschriebenen Triggererzeugung aussieht. Abbildung 6.9 zeigt im oberen Teil des Diagramms die logischen Pegel der Datenleitung zwischen Chipkartenterminal und Chipkarte. Man kann deutlich den ATR-Block, einen S-Block sowie die übertragenen Informationsblöcke erkennen. Wie im unteren Diagramm zu sehen ist, ist zwischen letztem Bit des Befehls und dem ersten Bit der Antwort das Triggersignal aktiviert. Genau in dieser Zeit wird von der Chipkarte die kryptographische Operation ausgeführt. Die tatsächliche Berechnungszeit wird etwas kürzer sein, da in diesem Zeitabschnitt der Befehl dekodiert werden muss und eine Antwort-APDU erzeugt werden muss. Deshalb wird diese Zeit im folgenden nicht Berechnungszeit, sondern Antwortzeit genannt.

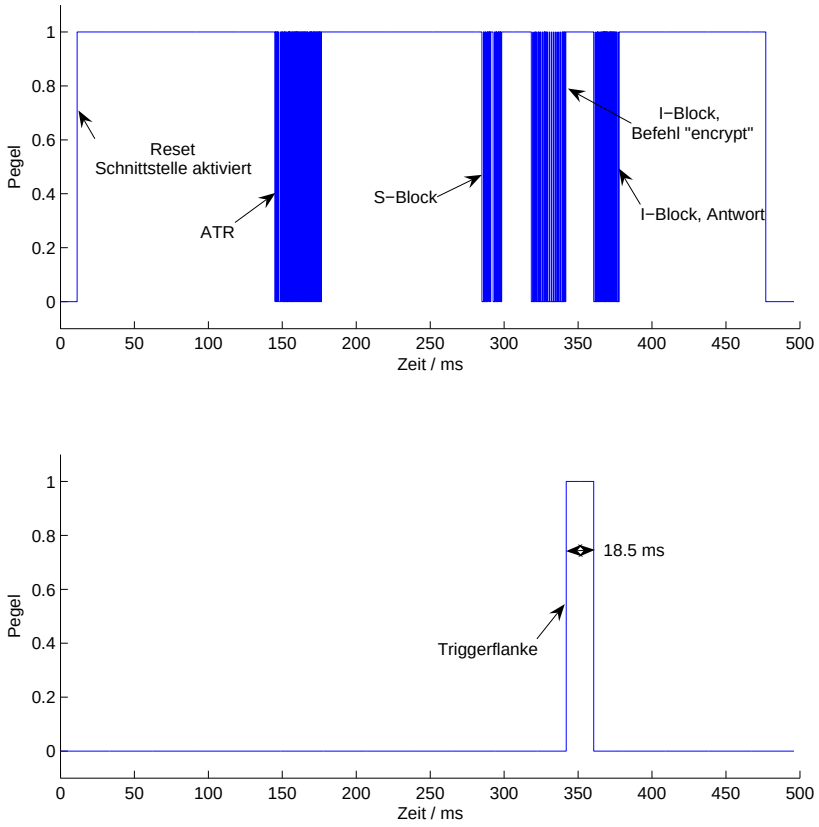

Abbildung 6.9: Oszillogramm der Datenkommunikation zwischen Terminal und Chipkarte sowie des generierten Triggersignals

6.3. Untersuchte Chipkarten

Im Rahmen dieser Arbeit wurden zwei BasicCards[1] der Firma Zeitcontrol[2] und eine Starcos-Karte von Giesecke und Devrient[3] auf ihre Seitenkanalresistenz untersucht. Dabei wurde versucht, die DES-Implementierungen der Karten mit einer DPA-Attacke zu brechen. Tabelle 6.1 gibt einen Überblick über die Kenndaten der untersuchten Chipkarten.

BasicCards sind low-cost Chipkarten, die sich in der Programmiersprache Basic programmieren lassen. Die Entwicklungsumgebung ist frei verfügbar, und die Karten kann man in kleinen Stückzahlen erwerben. Somit eignet sich die BasicCard in kleinen, geschlossenen Systemen beispielsweise für die Bezahlung oder für die Zugangskontrolle. BasicCards gibt es mit verschiedenen Ausstattungen von sehr einfachen Karten mit 1 KB EEPROM ohne Filesystem bis hin zu Karten mit über 30 KB nutzbarem Speicher und Coprozessoren für die Berechnung von asymmetrischen Verschlüsselungsverfahren. Die Karte mit der Bezeichnung ZC 3.3 ist ein älteres Modell, das nicht mehr vertrieben wird. Die ZC 4.5D Rev. F ist ein aktuelles Modell, das einen Coprozessor für die Berechnung von RSA besitzt.

Starcos ist ein Chipkartenbetriebssystem für professionelle Anwendungen. Die Entwicklungsumgebung ist im Gegensatz zur BasicCard teuer und erfordert größere Kenntnisse für die Entwicklung einer Anwendung. Von der Starcos-Karte sollte man erwarten, dass sie nicht anfällig für Seitenkanalangriffe ist. Da es sich jedoch um ein älteres Modell handelt, besteht die Möglichkeit, dass keine Gegenmaßnahmen implementiert sind.

Wir hatten keine Informationen darüber, welche Prozessoren auf den Chipkarten verwendet werden. Es handelt sich bei allen drei Karten vermutlich um Prozessoren von Philips. Bei den BasicCards wäre es denkbar, dass Karten mit der gleichen Bezeichnung Prozessoren von verschiedenen Herstellern verwenden, da das eigentliche Programm, das in der Programmiersprache Basic geschrieben wurde, in einen P-Code übersetzt wird, der in einer virtuellen Maschine unabhängig von der darunter liegenden Hardware ausgeführt wird.

Tabelle 6.1: Gegenüberstellung der untersuchten Chipkarten

Bezeichnung	ZC 3.3	ZC 4.5D Rev. F	Starcos SPK 2.2
EEPROM	8 KByte	30 KByte	k. a.
RAM	256 Byte	1 KByte	k. a.
Protokoll	T=1	T=0, T=1	T=0, T=1
Symmetrische Verschlüsselung	DES/Software	DES/Software	DES/Software
Asymmetrische Verschlüsselung	-	RSA 1024	RSA 1024
Floating-Point Arithmetik	Ja	Ja	k. a.
Filesystem	Ja	Ja	Ja

6.3.1. Das Testprogramm in den BasicCards

Das Testprogramm (siehe Listing 6.1), das in den Karten abläuft, hat lediglich die Aufgabe, bei Empfang eines entsprechenden Befehls in Form einer APDU (siehe Abbildung 6.4) eine

[1] http://www.basiccard.de
[2] http://www.zeitcontrol.com
[3] http://www.gdm.de

DES-Berechnung auszuführen und das Ergebnis an das Terminal zurückzusenden.

In Zeile eins des Programms wird festgelegt, dass das T=1 Protokoll verwendet werden soll. In den Zeilen drei bis fünf werden Schlüssel erzeugt. Wenn kein Parameter bei der Deklarierung angegeben wird, so wird ein Schlüssel, der ausschließlich aus Nullen besteht, angelegt. In den Zeilen sieben bis neun wird der eigentliche Verschlüsselungsbefehl festgelegt. Nach Angabe des Schlüsselwortes `Command` wird die Befehlsklasse (CLA) 0x80 angegeben, das ist die Klasse, die nach ISO 7816 für benutzerspezifische Befehle reserviert ist. Zusätzlich ist in diesem Byte codiert, dass kein Secure Messaging verwendet werden soll. Die eigentliche Befehlskennung (INS) wird als zweiter Parameter mit 0x01 angegeben. Nach dem Namen des Unterprogramms wird das neun Byte lange Daten-Feld der APDU, in Klammern gesetzt, festgelegt. Das Daten-Feld besteht aus einer Schlüsselnummer und dem zu verschlüsselnden Datenblock. Der eigentliche Aufruf der DES-Operation erfolgt in Zeile acht. Das Ergebnis der Berechnung wird im selben Format wie im Daten-Feld der Kommando-APDU an das Terminal zurückgesendet. In den Zeilen 11 bis 13 wird analog eine APDU für die Entschlüsselung eines DES-Chiffretexts definiert.

Abbildung 6.10 zeigt den Aufbau einer APDU, die eine DES-Verschlüsselungsoperation auf einer der beiden programmierten BasicCards startet. Die Felder CLA und INS wurden im letzten Absatz besprochen und sind dementsprechend gesetzt. Die Parameter P1 und P2 werden vom Basic-Programm nicht verwendet und sind auf den Wert Null gesetzt. Lc (length command) und Le (length expected) sind entsprechend der Länge des Datenwortes auf 9 gesetzt. Das Datenfeld besteht aus einer 1 Byte langen Schlüsselnummer und den darauf folgenden Daten mit der Länge von 8 Byte.

CLA	INS	P1	P2	Lc–Feld	Datenfeld	Le–Feld
80	01	00	00	09	Schlüssel (1 Byte) Datenblock (8 Byte)	09

Abbildung 6.10: APDU, welche die DES-Verschlüsselung auf der BasicCard startet

Listing 6.1: Einfaches Basicprogramm, das das Ausführen von DES-Operationen auf den BasicCards ermöglicht

```
1  #Pragma T = 1
2
3  Declare Key 1
4  Declare Key 2 = &Haf, &Hfe, &H08, &H91, &H82, &H58, &H8b, &Hbb
5  Declare Key 3 = &H01, &H23, &H45, &H67, &H89, &Hab, &Hcd, &Hef
6
7  Command &H80 &H01 desencrypt ( keynr As Byte, data As String*8 )
8      data = DES(+1, keynr, data)
9  End Command
10
11 Command &H80 &H02 desdecrypt ( keynr As Byte, data As String*8 )
12     data = DES(-1, keynr, data)
13 End Command
```

6.3.2. Das Testprogramm in der Starcos-Karte

Das Testprogramm in der Starcos-Karte ist eine Funktion des Starcos-Betriebssystems, die in der Entwicklungsumgebung aktiviert wurde. Der Wert `0x3132333435363738` wurde als Schlüssel verwendet. Die APDU zum verwendeten Befehl ist in Abbildung 6.11 dargestellt. Im Gegensatz zur APDU der BasicCards werden bei dieser APDU die Felder P1 und P2 verwendet. Das Feld P1 legt fest, im welchen Modus die DES-Operation arbeiten soll. Der hier angegebene Hex-Wert `00` steht für eine Verschlüsselung von einem einzigen Datenblock. Im Feld P2 wird angegeben, welcher Schlüssel verwendet werden soll. Der Hex-Wert `81` steht für den oben genannten Schlüssel.

CLA	INS	P1	P2	Lc–Feld	Datenfeld	Le–Feld
80	F8	00	81	08	Datenblock (8 Byte)	09

Abbildung 6.11: APDU, welche die DES-Verschlüsselung auf der Starcos-Karte startet

6.4. Messergebnisse

6.4.1. Antwortzeit

Die drei untersuchten Karten haben sehr unterschiedliche Antwortzeiten, obwohl die Taktfrequenz, die vom Chipkartenleser vorgegeben wird, bei allen Karten 3,57 MHz beträgt. Es ist anzunehmen, dass die beiden älteren Karten (BasicCard ZC 3.3 und Starcos SPK 2.2) diesen vorgegebenen Takt als Prozessortakt verwenden. Die neuere BasicCard (4.5D Rev. F) besitzt höchst wahrscheinlich eine PLL-Schaltung zur Frequenzvervielfachung. Das breitere Spektrum dieser Karte deutet ebenfalls darauf hin.

Tabelle 6.2: Antwortzeiten der analysierten Chipkarten

Bezeichnung	Antwortzeit
ZC 3.3	18,5 ms
ZC 4.5D Rev. F	5,4 ms
Starcos SPK 2.2	70,4 ms

6.4.2. Charakterisierung der Stromaufnahme

Wie in Abbildung 6.12 zu sehen ist, kann man an der Stromaufnahme der BasicCard ZC 3.3 die 16 Rundenfunktionen einer DES-Operation deutlich erkennen. Die Daten, die diesem Diagramm zugrunde liegen, wurden nicht nachbearbeitet und wurden mit einer Abtastrate von 25 $^{MS}/_{s}$ aufgezeichnet. Die BasicCard ZC 4.5 weist eine sehr ähnliche Stromaufnahme auf. Das deutet darauf hin, dass die gleiche Softwareimplementierung des DES-Algorithmus verwendet wird, die jedoch auf schnellerer Hardware ausgeführt wird. In der Stromaufnahme der

Starcos-Karte kann man keine charakteristischen Muster erkennen. Das könnte ein Hinweis auf mögliche Seitenkanal-Gegenmaßnahmen sein.

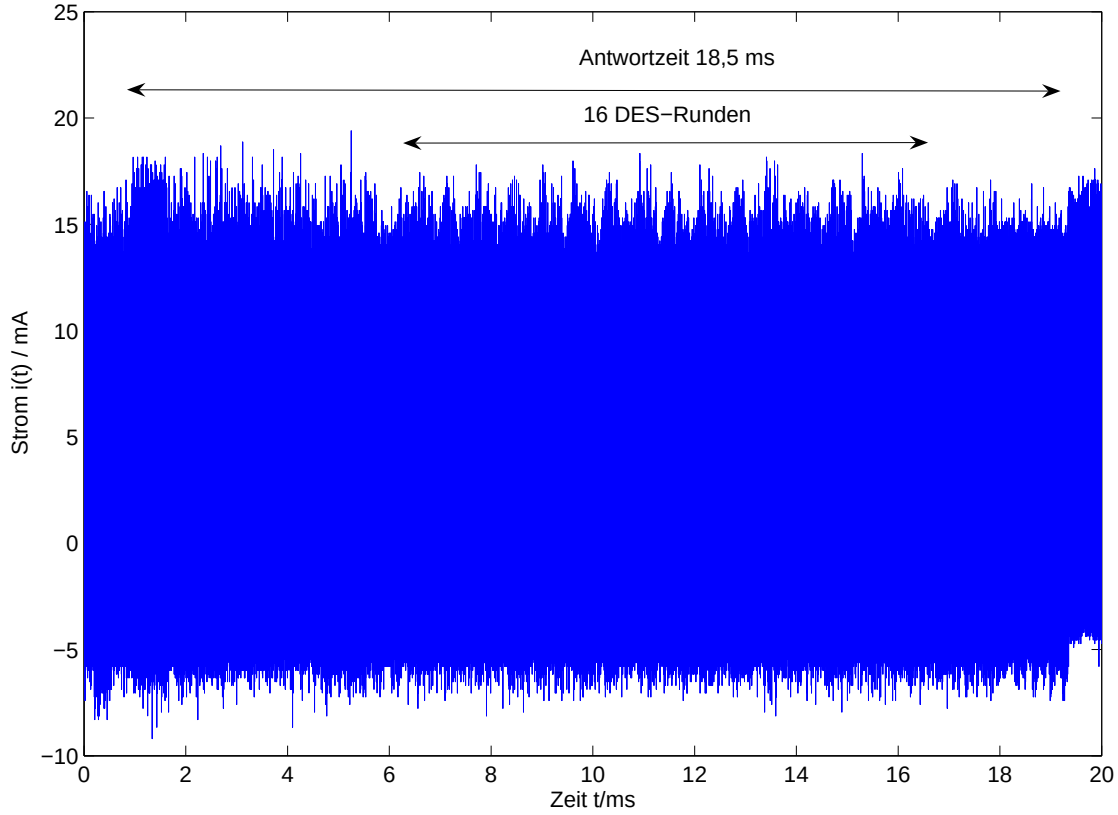

Abbildung 6.12: Stromaufnahme der BasicCard ZC 3.3 während der Berechnung einer DES-Operation. Die Daten für dieses Diagramm stammen direkt vom Oszilloskop und wurden nicht mit Methoden der Signalverarbeitung verändert. Die 16 Rundenfunktionen sind deutlich zu erkennen.

Die Stromaufnahme der drei untersuchten Karten weist jeweils ein charakteristisches Spektrum auf (siehe Abbildung 6.13 und Abbildung 6.14). Die beiden älteren Karten (BasicCard ZC 3.3 und Starcos SPK 2.2) zeigen weniger hochfrequente Anteile im Spektrum als die neuere BasicCard 4.5D. Das lässt vermuten, dass die neuere Karte eine PLL-Schaltung zur Erhöhung der Taktfrequenz des Prozessors eingebaut hat. Das würde auch die kürzere Antwortzeit dieser Chipkarte erklären. Die Amplitude der Spektren ist nicht auf eine physikalische Größe normiert.

6.4.3. Angriff auf eine exemplarische S-Box

In einem Experiment soll nun die erste S-Box des DES-Algorithmus auf den verschiedenen Chipkarten angegriffen werden. Bei den Angriffen auf die zwei BasicCards kam der gleiche Schlüssel zum Einsatz. Der zugehörige angegriffene Teilschlüssel ist 38. Auf der Starcos-Karte kam ein anderer Schlüssel mit dem korrespondierenden Teilschlüssel 20 zum Einsatz. Angegriffen wurde in allen Fällen das erste Target-Bit.

In Abbildung 6.15 sind die Ergebnisse des Angriffs auf die BasicCard ZC 3.3 dargestellt. In

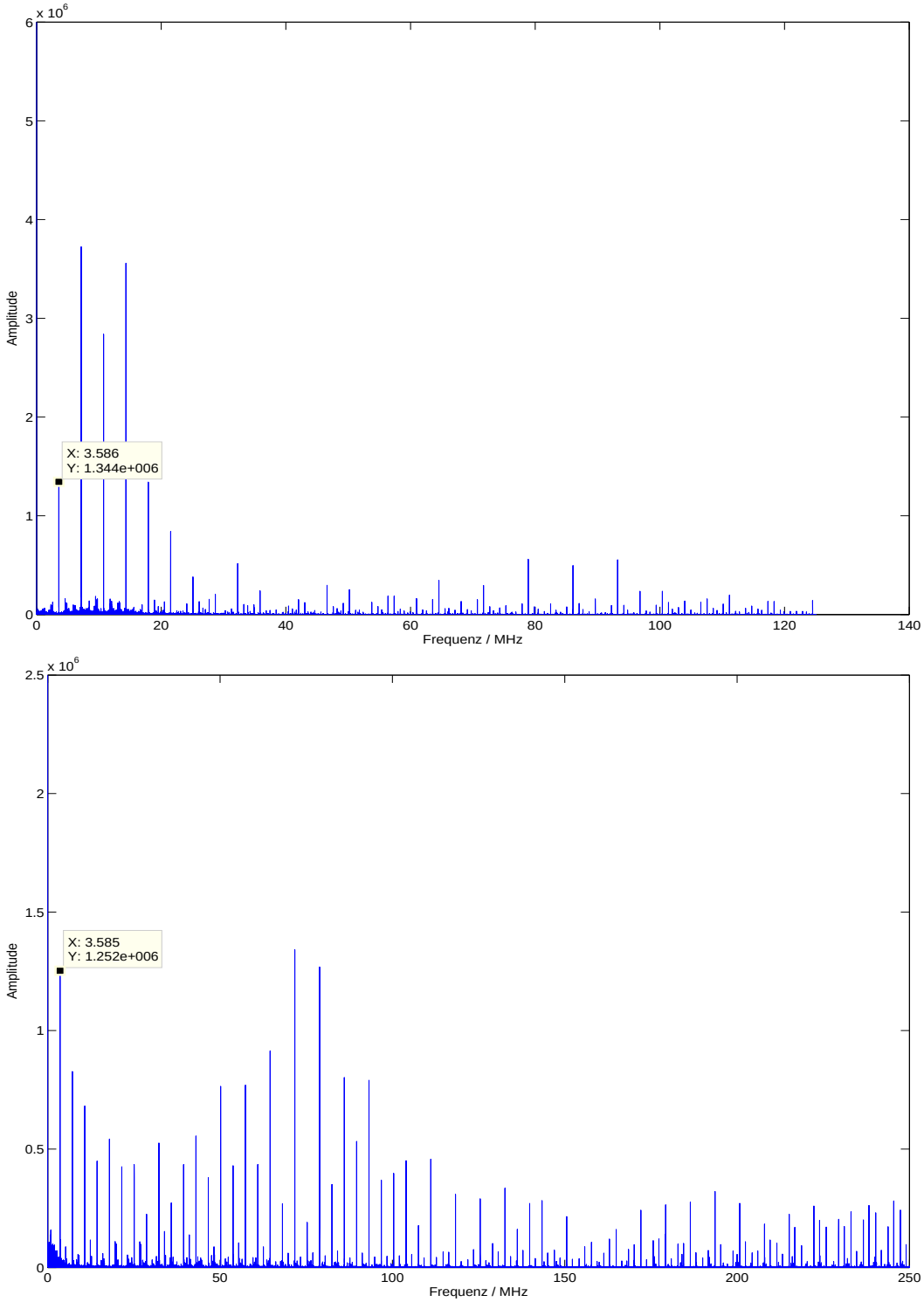

Abbildung 6.13: Das Spektrum der Stromaufnahme der Chipkarten BasicCard ZC 3.3 (oben), BasicCard ZC 4.5D (unten).

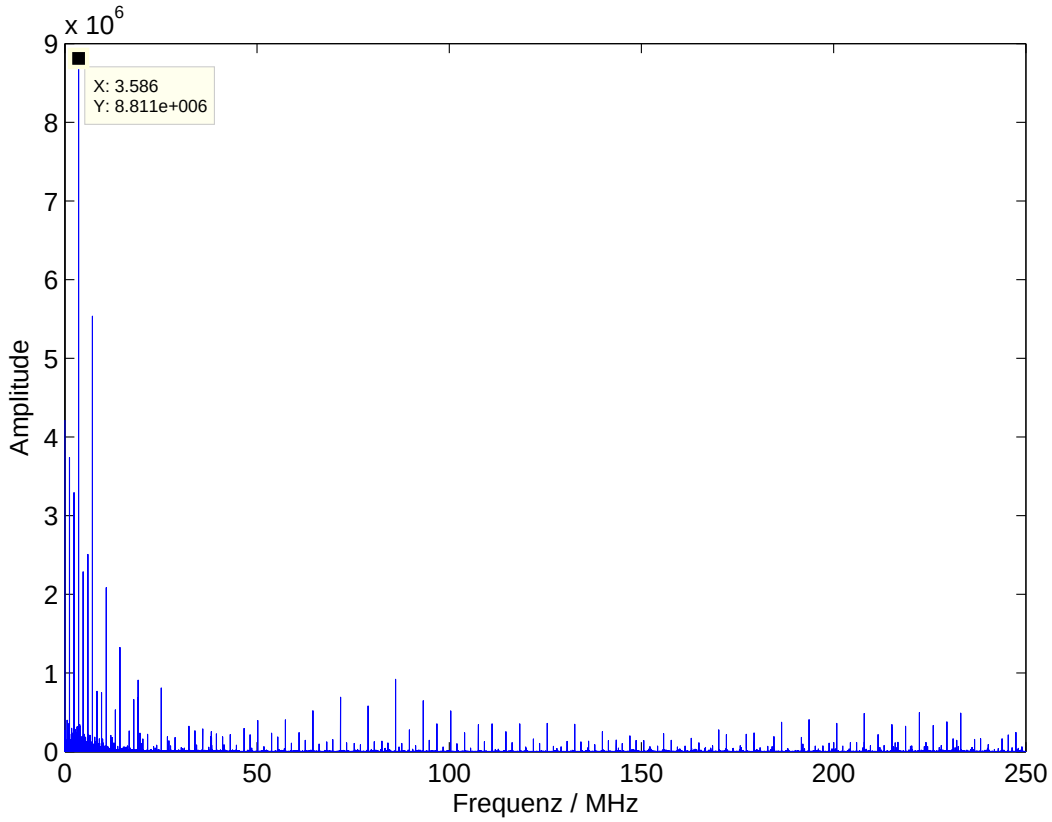

Abbildung 6.14: Das Spektrum der Stromaufnahme der Starcos SPK 2.2-Karte.

der oberen Hälfte der Abbildung sind 200 Messkurven in die Berechnung eingeflossen. Im Gegensatz dazu wurden in der unteren Hälfte 2000 Messkurven aus dem gleichen Messdatensatz verwendet. Die Messdaten wurden mit einer Abtastfrequenz von 250 $^{\mathrm{MS}}/_{\mathrm{s}}$ aufgezeichnet (= 70-faches Oversampling) und nach dem zeitlichen Ausrichten einem Downsampling mit dem Faktor 10 unterzogen. In der linken Hälfte ist jeweils die Korrelationskurve der wahrscheinlichsten Schlüsselhypothese zu sehen, die sich auch mit dem richtigen Teilschlüssel deckt. In der rechten Hälfte sind die Absolutwerte der Spitzenwerte der jeweiligen Korrelationskurven aufgetragen. An den beiden Korrelationskurven ist zu sehen, wie sich die Spitzen mit steigender Anzahl von Messungen zunehmend vom Rauschen abheben. Man kann jedoch auch sehen, dass die 10-fache Anzahl von Messkurven das Ergebnis der Auswertung nicht in großem Maße verbessert.

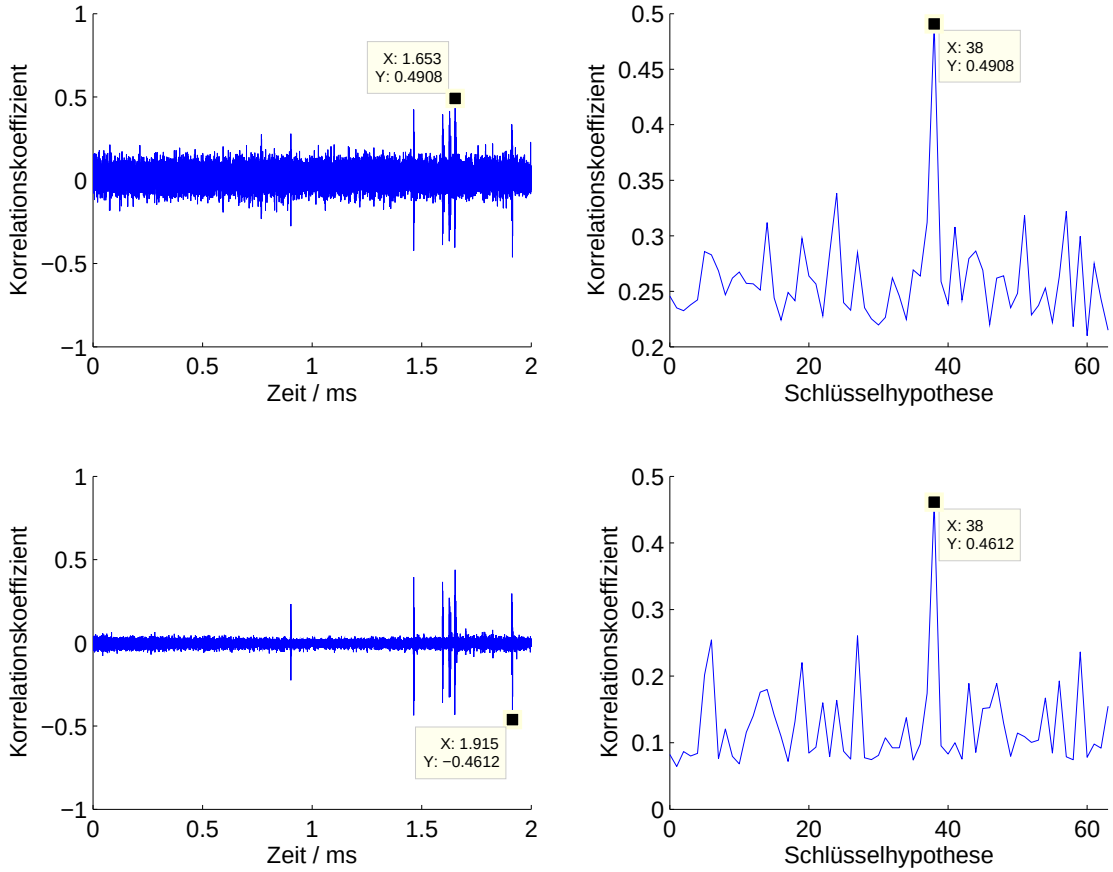

Abbildung 6.15: Ergebnisse einer DPA-Attacke auf die BasicCard ZC 3.3 mit 200 Messkurven (oben) und 2000 Messkurven (unten). Links ist die wahrscheinlichste Korrelationskurve und rechts sind die Absolutwerte der Spitzenwerte aller 64 Korrelationskurven bzw. Schlüsselhypothesen dargestellt.

In Abbildung 6.16 sind die wahrscheinlichsten Korrelationskurve und die absoluten Spitzenwerte aller Korrelationskurven für die BasicCard ZC 4.5D dargestellt. Hier wurden 500 Messkurven ($N = 500$) für die Analyse verwendet. Die Messkurven wurden mit einer Abtastfrequenz von 500 $^{\mathrm{MS}}/_{\mathrm{s}}$ aufgezeichnet (= 140-faches Oversampling) und nach dem zeitlichen Ausrichten einem Downsampling mit dem Faktor 10 unterzogen. Bei dieser Chipkarte hebt

sich das Maximum nicht mehr so deutlich vom Rauschen ab. Es werden also in der Regel mehr Messungen als bei der BasicCard ZC 3.3 benötigt.

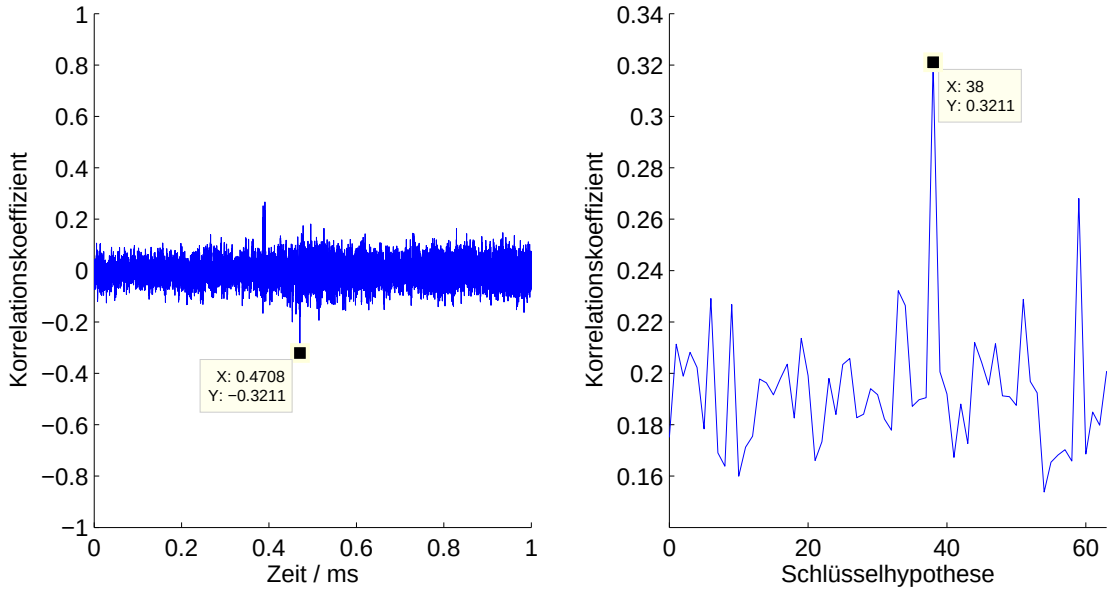

Abbildung 6.16: Ergebnisse einer DPA-Attacke auf die BasicCard ZC 4.5D mit 500 Messkurven. Links ist die wahrscheinlichste Korrelationskurve und rechts sind die Absolutwerte der Spitzenwerte aller 64 Korrelationskurven bzw. Schlüsselhypothesen dargestellt.

Ein Angriff auf die Starcos-Karte war in der zur Verfügung stehenden Zeit von drei Tagen nicht erfolgreich. Die große Antwortzeit und die gleichförmige Stromaufnahme hat es verhindert, den Zeitraum einer DES-Runde mit ausreichend hoher Abtastrate aufzuzeichnen. Das Fehlen von charakteristischen Mustern in der Stromaufnahme hat das zeitliche Ausrichten der Kurven erschwert. Darüber, ob wirksame DPA-Gegenmaßnahmen implementiert sind, kann keine Aussage getätigt werden, da man dafür noch mehr Zeit in DPA-Analysen investieren müsste.

6.4.4. Anzahl der benötigten Messungen

In diesem Abschnitt soll nun versucht werden, die Anzahl der benötigten Messungen für einen erfolgreichen Angriff für alle verwendeten Chipkarten zu ermitteln. Die Anzahl der Messungen hängt allerdings sehr stark vom verwendeten Schlüssel und von den angegriffenen S-Boxen und Target-Bits ab. Vor allem die Abhängigkeit von der angegriffen Struktur ist sehr ausgeprägt, weshalb es schwer ist, eine allgemein gültige Aussage zu treffen. Im folgenden werden die gleichen Analysen bei zwei verschiedenen Schlüsseln durchgeführt, um die Abhängigkeit vom geheimen Schlüssel zu demonstrieren.

In Abbildung 6.17 wird die Anzahl der benötigten Messungen bei der Attacke einer BasicCard 3.3 dargestellt. Das linke Diagramm wurde mit dem Schlüssel `0xaffe089182588bbb` und das rechte mit dem Schlüssel `0x0123456789abcdef` ermittelt. Auf der Abszisse ist die angegriffene S-Box aufgetragen, die jeweils vier Target-Bits hat. Auf der Ordinate ist die Anzahl der Messungen aufgetragen, die mindestens nötig sind, damit man den jeweiligen Teilschlüssel

richtig ermitteln kann.

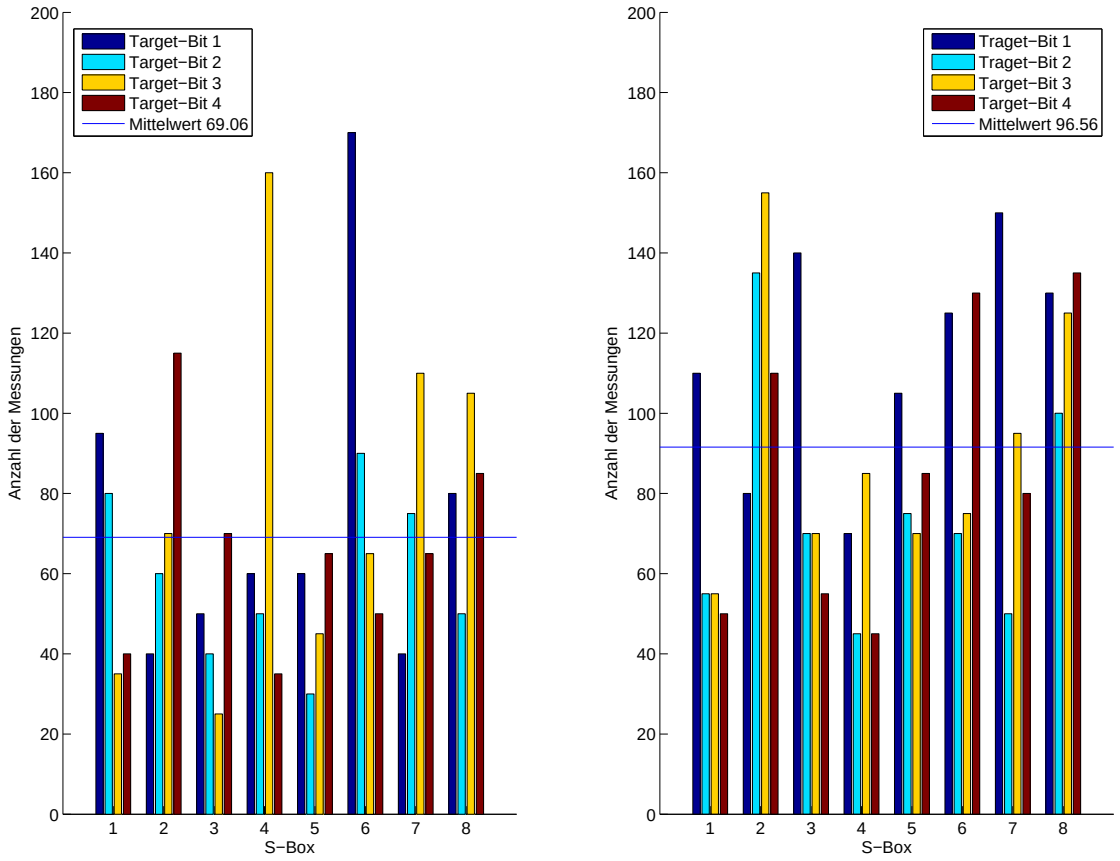

Abbildung 6.17: Anzahl der benötigten Messkurven für einem erfolgreichen DPA-Angriff mit der Korrelationsmethode. Die Balken stehen dabei für die acht S-Boxen mit den jeweils vier möglichen Target-Bits. Links wurde der Schlüssel `0xaffe089182588bbb` und rechts der Schlüssel `0x0123456789abcdef` verwendet.

In Abbildung 6.18 werden die benötigten Messungen für die Attacke einer BasicCard 4.5D dargestellt. Wie bei der BasicCard ZC 3.3 wurden auch hier die Schlüssel `0xaffe089182588bbb` und `0x0123456789abcdef` für das linke und rechte Diagramm verwendet. Wie schon im vorigen Abschnitt beim Angriff auf die exemplarische S-Box vorhergesagt wurde, sind für einen erfolgreichen Angriff auf diese Karte mehr Messungen nötig als für die BasicCard ZC 3.3.

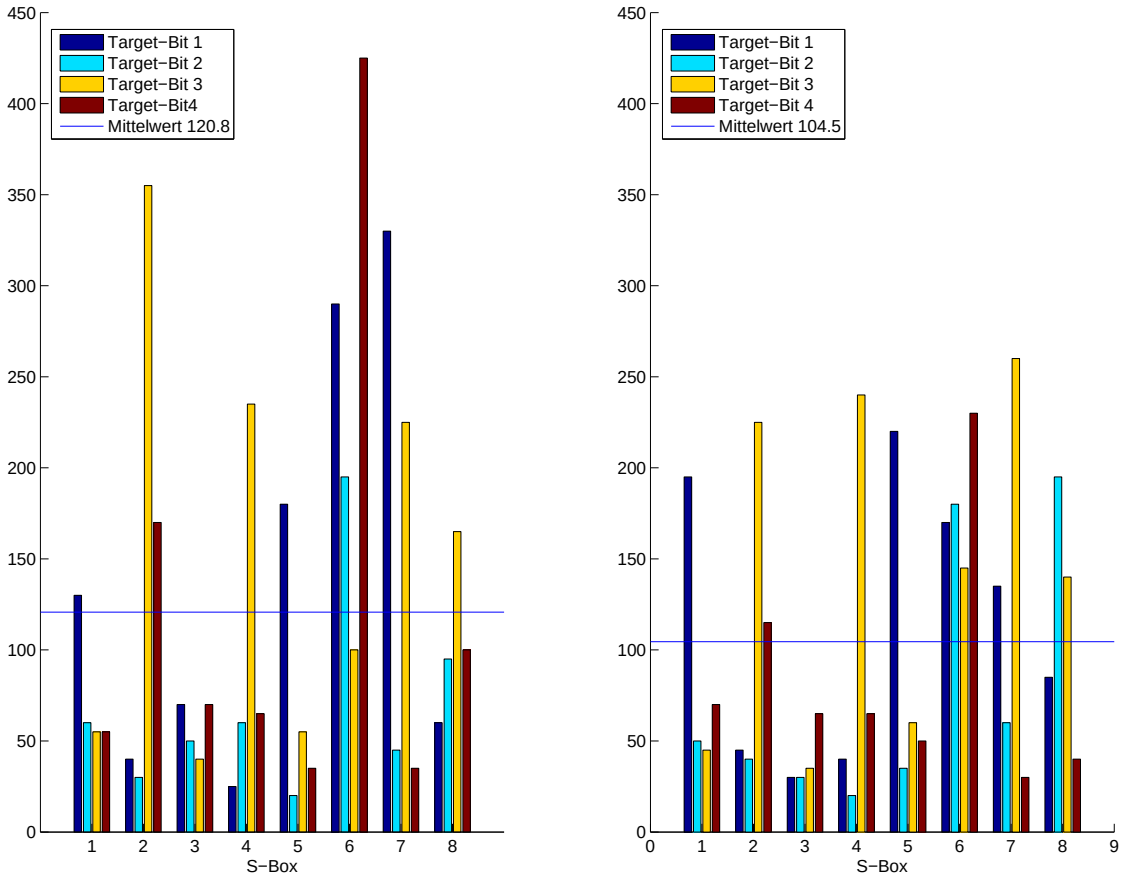

Abbildung 6.18: Anzahl der benötigten Messkurven für einen erfolgreichen DPA-Angriff mit der Korrelationsmethode. Die Balken stehen dabei für die acht S-Boxen mit den jeweils vier möglichen Target-Bits. Links wurde der Schlüssel `0xaffe089182588bbb` und rechts der Schlüssel `0x0123456789abcdef` verwendet.

Kapitel 7.

Zusammenfassung und Ausblick

In dieser Diplomarbeit wurden DPA-Attacken auf verschiedene Implementierungen symmetrischer kryptographischer Algorithmen durchgeführt. Als Zielplattformen kamen drei verschiedene Hardwarearchitekturen zum Einsatz. Zwei davon waren herkömmliche Mikroprozessoren, wie sie in großer Zahl in Embedded Systems zum Einsatz kommen. Bei diesen Systemen hat man große Freiheit, was den Aufbau einer Schaltung und die Entwicklung der Software betrifft. Den Startzeitpunkt für die Messung mittels Oszilloskop kann man sehr exakt durch Setzen eines Triggersignals festlegen. Unter diesen idealen Voraussetzungen konnte gezeigt werden, das bei entsprechend störungsresistentem Aufbau ab ca. 50 Messungen bei einer gängigen Software-DES-Implementierung erfolgreiche DPA-Angriffe möglich sind. Es konnte auch gezeigt werden, dass bei einer DPA-Attacke ein Schirmgehäuse nur eine minimale Verkleinerung der Anzahl der benötigten Messungen bringt und im schlechtesten Fall sogar eine kleine Erhöhung der Anzahl zur Folge hat. Diese Erhöhung ist auf Störungen innerhalb des Gehäuses, hervorgerufen durch Kopplungen zwischen Leitern, zurückzuführen.

Als dritte Hardwareplattform wurden verschiedene Chipkarten analysiert. Bei den Chipkarten lag der Schwerpunkt beim Erzeugen eines geeigneten Triggersignals für das Oszilloskop. Spezielle Entstörungsmaßnahmen, wie die galvanische Trennung von Chipkarte und Terminal, wurden nicht vorgenommen. Die Problematik beim Analysieren von Chipkarten ist die exakte zeitliche Ausrichtung der einzelnen Stromkurven. Damit dies funktioniert, ist eine hohe Abtastrate des Oszilloskops nötig, was in großen Mengen von Messdaten resultiert. Außerdem ist es nötig, ein charakteristisches Muster zu finden, welches in jeder Stromkurve an der gleichen Stelle vorkommt. Trotz dieser Schwierigkeiten ist es gelungen, den DES-Schlüssel einer älteren und einer aktuellen BasicCard mittels DPA-Attacke zu ermitteln.

Mit der in dieser Diplomarbeit weiterentwickelten Software und dem entwickelten Messaufbau ist es möglich, einen kompletten DES oder AES Schlüssel in ca. einem halben Tag zu ermitteln. In diese Zeit ist die Vorbereitung und das Aufnehmen von Messdaten schon eingerechnet. Die eigentliche DPA-Attacke mit bereits gemessenen Daten dauert lediglich ein paar Minuten. Ein größerer Schlüsselraum erhöht den zeitlichen Aufwand nur minimal, da der Schlüssel „Stück für Stück“ rekonstruiert wird.

Für weiterführende Arbeiten wäre es interessant, sich mit dem Pattern-Matching für die zeitliche Ausrichtung von Stromkurven zu beschäftigen. Bei zu geringer Abtastrate funktioniert das Pattern-Matching nicht bei jeder Stromkurve, was in einem Verwerfen der Stromkurve resultiert. Es wäre auch wünschenswert, wenn die entsprechenden Suchfenster automatisch von einem Skript ermittelt werden und nicht vom Benutzer manuell eingestellt werden müssten. Weitere interessante Themen wären die Ausweitung der Untersuchung auf Hardwareimplementierungen von symmetrischen kryptographischen Algorithmen und das Beschäftigen mit asymmetrischer Kryptographie.

Anhang A.

Fertigungsunterlagen

Dieser Anhang enthält die Fertigungsunterlagen für die während der Diplomarbeit entstandenen Platinen. Für das Erstellen der Fertigungsunterlagen wurde die Software Eagle 4.13 von CadSoft[1] verwendet. Layout und Bestückungsplan sind im Maßstab 1:1 abgebildet. Die Platinen besitzen jeweils eine Kupfer-Leiterstruktur. Nicht vermeidbare Leitungskreuzungen sind als Drahtbrücken ausgeführt, die in Bestückungsplänen als Linie eingezeichnet sind. Alle Layouts und Bestückungspläne sind von oben gesehen dargestellt. Beim Layout des Messobjekts sieht man an der spiegelverkehrten Beschriftung, dass sich die Kupferschicht auf der Unterseite der Platine befindet. Beim Layout der Triggererzeugung befindet die die Kupferschicht dagegen auf der Oberseite. Auf den Bestückungsplänen sind Ober- und Unterseite gemeinsam dargestellt. Die bedrahteten Bauelemente sind jeweils auf der Oberseite zu bestücken, die SMD-Elemente auf der Seite der Kupferschicht.

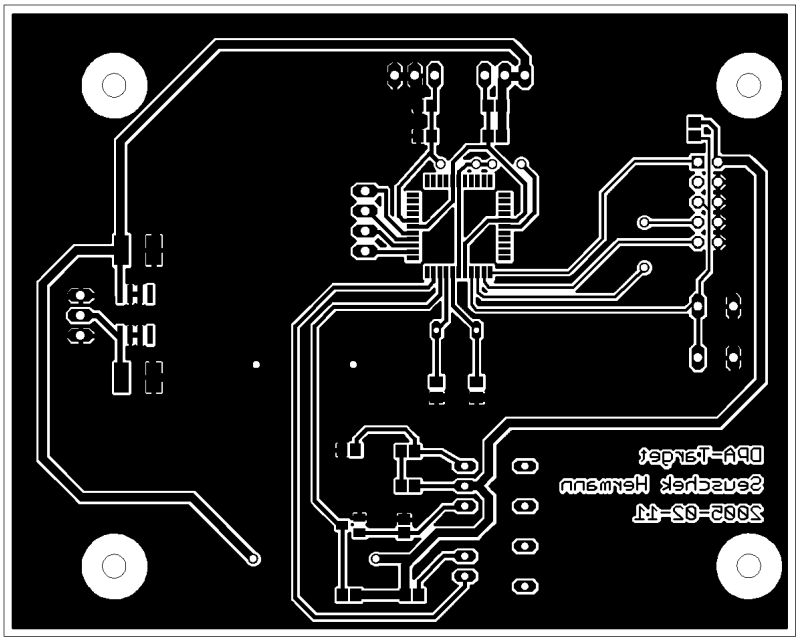

Abbildung A.1: Platinenlayout des ATmega32 Messobjekts

[1] http://www.cadsoft.de

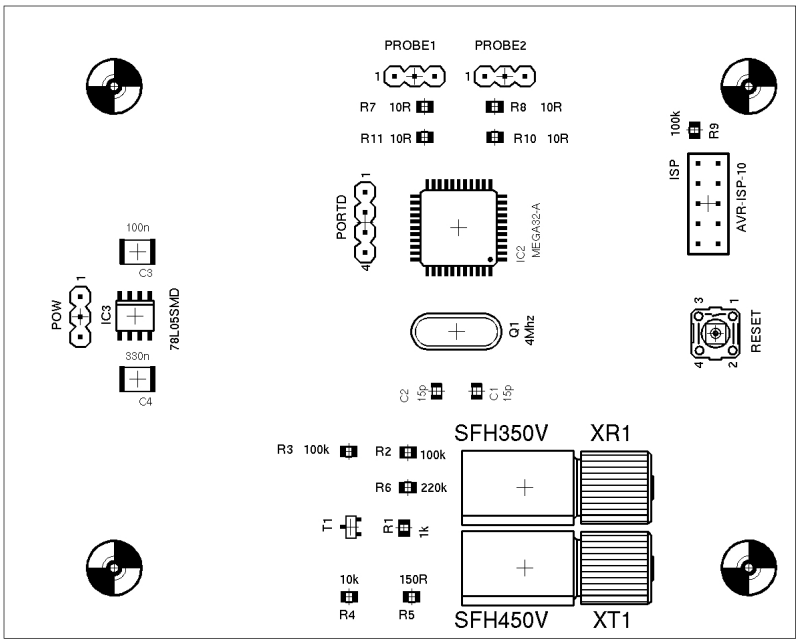

Abbildung A.2: Bestückungsplan des ATmega32 Messobjekts

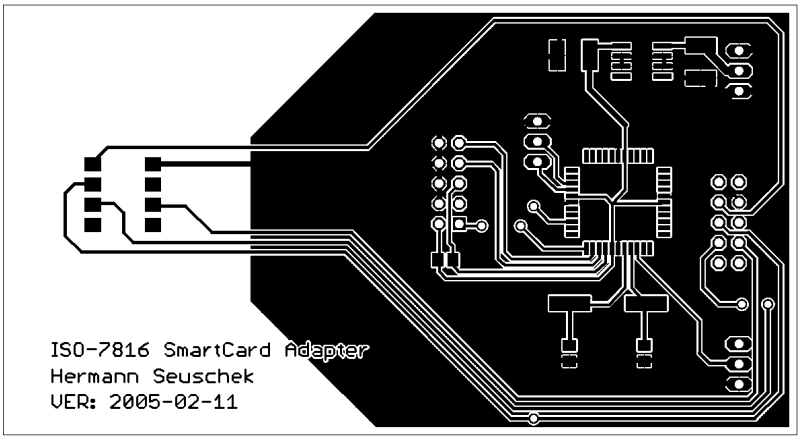

Abbildung A.3: Platinenlayout der Triggererzeugung

Tabelle A.1: Bauteilliste des ATmega32 Messobjekts

Name	Wert	Bauteil	Gehäuse
C1	15p	C-EUC0805K	C0805K
C2	15p	C-EUC0805K	C0805K
C3	100n	C-EUC1812K	C1812K
C4	330n	C-EUC1812K	C1812K
IC2	MEGA32-A	MEGA32-A	TQFP44
IC3	78L05SMD	78L05SMD	SO08
ISP	AVR-ISP-10	AVR-ISP-10	AVR-ISP-10
PORTD		MA04-1	MA04-1
POW		MA03-1	MA03-1
PROBE1		MA03-1	MA03-1
PROBE2		MA03-1	MA03-1
Q1	4MHz	XTAL/S	QS
R1	1k	R-EU_R0805	R0805
R2	100k	R-EU_R0805	R0805
R3	100k	R-EU_R0805	R0805
R4	10k	R-EU_R0805	R0805
R5	150R	R-EU_R0805	R0805
R6	220k	R-EU_R0805	R0805
R7	10R	R-EU_R0805	R0805
R8	10R	R-EU_R0805	R0805
R9	100k	R-EU_R0805	R0805
R10	10R	R-EU_R0805	R0805
R11	10R	R-EU_R0805	R0805
RESET		10-XX	B3F-10XX
T1		-NPN-SOT23-EBC	SOT23-EBC
XR1	SFH350V	SFH350V	SFH-V1
XT1	SFH450V	SFH450V	SFH-V

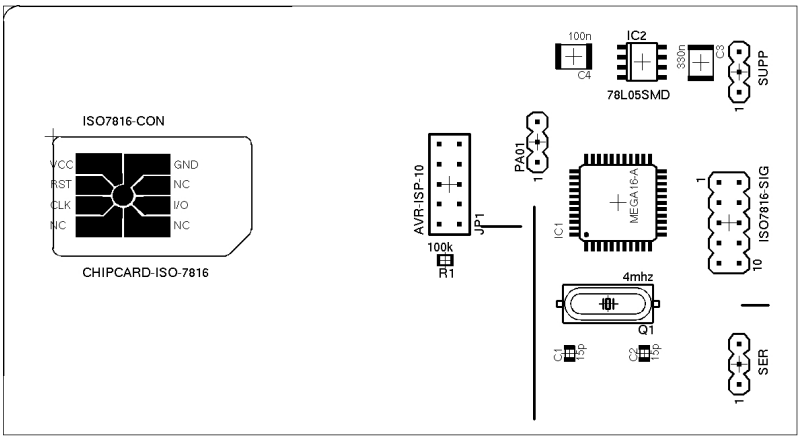

Abbildung A.4: Bestückungsplan der Triggererzeugung

Tabelle A.2: Bauteilliste der Triggererzeugung

Name	Wert	Bauteil	Gehäuse
C1	15p	C-EUC0805K	C0805K
C2	15p	C-EUC0805K	C0805K
C3	330n	C-EUC1812K	C1812K
C4	100n	C-EUC1812K	C1812K
IC1	MEGA16-A	MEGA16-A	TQFP44
IC2	78L05SMD	78L05SMD	SO08
ISO7816-CON	CHIPCARD-ISO-7816	CHIPCARD-ISO-7816	ISO7816
ISO7816-SIG		MA05-2	MA05-2
JP1	AVR-ISP-10	AVR-ISP-10	AVR-ISP-10
PA01		MA03-1	MA03-1
Q1	4MHz	CRYTALSM49	SM49
R1	100k	R-EU_M0805	M0805
SER		MA03-1	MA03-1
SUPP		MA03-1	MA03-1

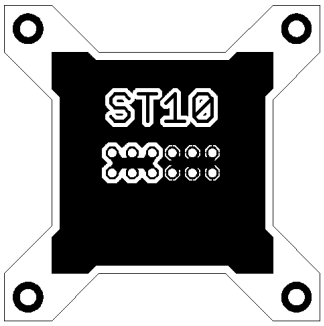

Abbildung A.5: Platinenlayout des ST10-Adapters

Literaturverzeichnis

[AG01] M. L. Akkar and Chr. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In C. K. Koç, D. Naccache, and Chr. Paar, editors, *Proceedings of CHES '2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer-Verlag, 2001. 16, 69

[AL05] AVR-LIBC. *AVR-Libc 1.2.3 Manual*, 2005. See http://www.nongnu.org/avr-libc/user-manual/. 53

[AO00] M. Aigner and E. Oswald. Power Analysis Tutorial. Technical report, IAIK, Graz, 2000. See http://www.iaik.tu-graz.ac.at/aboutus/people/oswald/. 28, 43

[ATM04] ATMEGA32. *ATmega32 Complete Datasheet*, 2004. See http://www.atmel.com/products/avr. 51

[BB03] D. Brumley and D. Boneh. Remote Timing Attacks are Practical. In *Proceedings of 12th USENIX Security Symposium 2003*, pages 1–14, 2003. See http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf. 6

[BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *Proceedings of EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997. 14

[DR99] J. Daemen and V. Rijmen. AES Proposal: Rijndael, September 1999. See http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf. 12, 69

[Fra02] J. Franz. *EMV – Störungssicherer Aufbau elektronischer Schaltungen.* Teubner, 2002. 22

[HJMS00] E. Hess, N. Janssen, B. Meyer, and T. Schütze. Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures—A Survey. In *Proceedings of EUROSMART Security Conference, June 2000, 13–15*, pages 55–64, Marseille, France, 2000. 3, 6

[JQ04] M. Joye and J.-J. Quisquater, editors. *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, August 11–13, 2004, Cambridge, MA, USA, 2004. Springer-Verlag. 3, 15

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *Proceedings of CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999. 3, 7

[KKP02] B. S. Kaliski, C. K. Koç, and Chr. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, August 13–15, 2002, Redwood Shores, California, USA, 2002. Springer-Verlag. 3, 15

[KNP01] C. K. Koç, D. Naccache, and Chr. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, May 14–16, 2001, Paris, France, 2001. Springer-Verlag. 3, 15

[Koc96] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Proceedings of CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996. 3, 6

[KP99] C. K. Koç and Chr. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, August 12–13, 1999, Worcester, MA, USA, 1999. Springer-Verlag. 3, 15

[KP00] C. K. Koç and Chr. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, August 17–18, 2000, Worcester, MA, USA, 2000. Springer-Verlag. 3, 15

[Man02] S. Mangard. Calculation and Simulation of the Susceptibility of Cryptographic Devices to Power-Analysis Attacks. Diplomarbeit, IAIK, TU Graz, January 2002. See http://www.iaik.tu-graz.ac.at/aboutus/people/mangard/. 10, 43

[Man04] S. Mangard. *Securing Implementations of Block Ciphers against Side-Channel Attacks*. PhD thesis, IAIK, TU Graz, August 2004. See http://www.iaik.tu-graz.ac.at/aboutus/people/mangard/. 7, 43

[Mes00a] T. S. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms*. PhD thesis, University of Illinois at Chicago, 2000. 234 pages. 21, 34

[Mes00b] T. S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *Proceedings of CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000. 71

[MvOV96] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996. See http://www.cacr.math.uwaterloo.ca/hac/. 11, 14, 55, 67

[NIS99] NIST. *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication 46-3, October 1999. Available at http://csrc.nist.gov/publications/fips. 11, 55, 67

[NIS01] NIST. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197, November 2001. Available at http://csrc.nist.gov/publications/fips. 12, 69

[Osw99] E. Oswald. Analyse der Anwendung von DPA auf DES-Bausteine. Diplomarbeit, IAIK, TU Graz, September 1999. See http://www.iaik.tu-graz.ac.at/aboutus/people/oswald/papers/dpl.pdf.

[Osw03] E. Oswald. *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*. PhD thesis, IAIK, TU Graz, May 2003. See http://www.iaik.tu-graz.ac.at/aboutus/people/oswald/. 43

[QK02] J. J. Quisquater and F. Koeune. State-of-the-art regarding side channel attacks, October 2002. See http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf. 3, 6

[RE02] W. Rankl and W. Effing. *Handbuch der Chipkarten*. Hanser, 2002. 78, 79

[Rod00] A. Rodewald. *Elektromagnetische Verträglichkeit*. Vieweg, 2000. 22, 24

[Sch00] W. Schindler. A Timing Attack against RSA with the Chinese Remainder Theorem. In *Proceedings of CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer-Verlag, 2000. 6

[STM02] STMicroelectronics. *ST10F168 Datasheet*, January 2002. See http://www.keil.com/dd/chip/3184.htm. 65

[WIK05] Mikrocontroller.net WIKI. STK200 – AVR In System Programmer, April 2005. See http://www.mikrocontroller.net/wiki/STK200 and http://www.mikrocontroller.net/wiki/AVR-ISP_FAQ. 53

[WKP03] C. D. Walter, C. K. Koç, and Chr. Paar, editors. *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, Workshop on Cryptographic Hardware and Embedded Systems, September 8–10, 2003, Cologne, Germany, 2003. Springer-Verlag. 3, 15