Technische Universität München
Fakultät für Elektro- und Informationstechnik
Lehrstuhl für Sicherheit in der Informationstechnik

SmartCard Projektpraktikum

Leitfaden zum Praktikum - Teil 1

Betreuer:	Dr.-Ing. Martin Strasser, M. Sc. Oscar Guillen
Betreuender Hochschullehrer:	Prof. Dr.-Ing. Georg Sigl
Tutor:	B.Sc. Robert Ingr

B.Sc. Robert Ingr

Inhaltsverzeichnis

Kapitel 1: Praktikumsübersicht

1.1 Aufgabensbeschreibung

Im Projektpraktikum SmartCard wird die Sicherheit eines exemplarischen ein PayTV-Systems untersucht und verbessert.

Das „PayTV-System“ besteht aus einem Server, welcher einen Videostream verschlüsselt überträgt. Er stellt den Sender eines Broadcastsystems dar. Auf Empfängerseite existiert eine SmartCard, die es ermöglicht das Videosignal zu entschlüsseln. Hierfür führt die SmartCard eine Entschlüsselung mit einem geheimen Schlüssel aus vgl. 1.1.

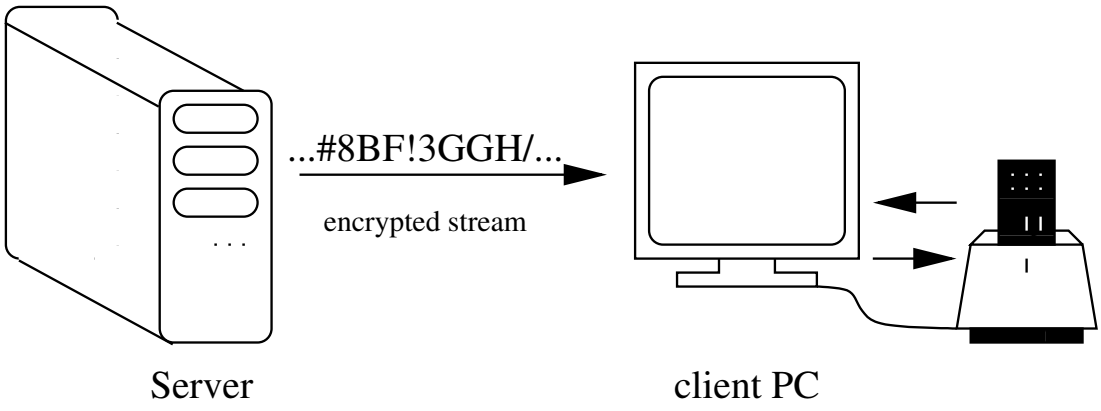


Abbildung 1.1: System Übersicht

Diesen Schlüssel gilt es aus der gegebenen Karte zu extrahieren. Der Angriff, den Sie durchführen, erfolgt über eine Seitenkanalanalyse. Der Seitenkanal soll der Stromverbrauch der Karte sein (DPA - Differential Power Analysis). Damit sollen Sie den ersten Schritt durchführen und das System kompromittieren.
Weiter soll das Ergebnis auch genutzt werden, und mit einer weiteren Karte der Stream entschlüsselt werden. Sie wollen also ein Duplikat der Karte anfertigen. Hierzu setzen Sie sich mit dem Aufbau und der Funktionalität von standard SmarCards auseinander. Anschließend implementieren Sie dies und programmieren eine „leere“ Karte mit dem zuvor geknackten Schlüssel.
Die beiden Aufgaben, Angriff und Implementierung der Klonkarte werden parallel gelöst.

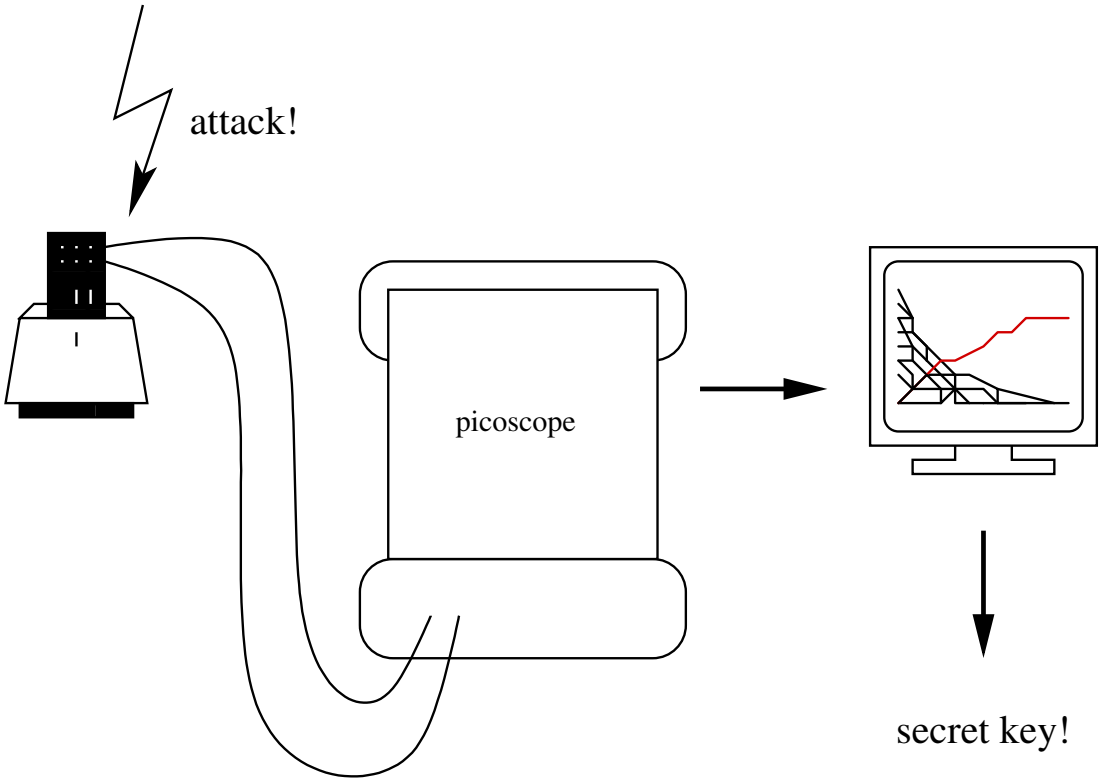


Abbildung 1.2: Measure

Ihre Aufgaben im ersten Praktikumsteil sind:

- DPA Angriff - Extraktion des Schlüssels
- Implementierung der Klonkarte

Im zweiten Teil des Praktikums werden Gegenmaßnahmen für DPA Angriffe erarbeitet. Nachdem Sie sich im ersten Praktikumsteil ein Verständnis für diesen Angriff angeeignet haben, können Sie sich nun Mechanismen überlegen, um den Angriff zu erschweren und Ihre Implementierung zu härten. Diese Mechanismen integrieren Sie in Ihre Klonkarte und wiederholen den Angriff. Es bietet sich an, den Angriff entsprechend Ihrer Gegenmaßnahmen zu modifizieren. Sie können hier noch einmal die Verarbeitung der Messdaten optimieren um einen effizienteren Angriff zu fahren. Ihr Ziel ist es schließlich, die entwickelten Gegenmaßnahmen hinsichtlich Aufwand und Nutzen zu bewerten. Auf eine übersichtliche Evaluierung der Gegenmaßnahmen wird besonderer Wert gelegt.

Ihre Aufgaben im zweiten Praktikumsteil sind:

- Gegenmaßnahmen zu DPA implementieren
- Methoden der DPA optimieren
- Gegenmaßnahmen evaluieren

1.2 Fähigkeiten

1.2.1 DPA

Um den Angriff durchzuführen, werden Sie den Stromverbrauchs des Chips messen und die Messdaten statistisch auswerten. Die Verarbeitung der Daten stellt im Wesentlichen die DPA Analyse dar. Hierzu können Sie beispielsweise mit Matlab oder Python arbeiten. Vorkenntnisse in diesen Programmiersprachen erleichtern die Aufgabe zwar, sind aber nicht zwingend notwendig. Das was Sie benötigen, können Sie sich mit wenig Zeitaufwand im Rahmen des Praktikums aneignen.

1.2.2 Duplikat SmartCard

Um diesen Aufgabenteil zu bewältigen, programmieren Sie üblicherweise in C (andere Methoden sind auch möglich). Sie entwickeln die Firmware für den Atmel Atmega 644. Die Entwicklungsumgebung können Sie sich frei auswählen (z.B. Kontrollerlab, Eclipse, AVRStudio...). Erfahrung mit der Entwicklung für Mikroprozessoren und eingebetteten Systemen sind von Vorteil, lassen sich aber auch im Rahmen des Praktikums aneignen. Kenntnisse in der Programmiersprache C sollten Sie für diesen Teil auf jeden Fall mitbringen.

1.3 Organisatorisches

1.3.1 Gruppenbildung, Teamaufteilung

Nach den Einführungsterminen bilden Sie Gruppen zur Bearbeitung des Projektpraktikums. Je nach Teilnehmerzahl werden Gruppen zu etwa vier Personen zusammenarbeiten. Jede Gruppe hat dann die Praktikumsaufgaben eigenständig zu erledigen. Für jede Gruppe existiert ein Videostreamserver und ein geheimer Schlüssel. Innerhalb der Gruppe empfehlen wir, zwei Teams (Team A, Team B) zu bilden. Team A übernimmt die Implementierung des Angriffs während Team B die Klonkarte implementiert. Achten Sie trotz der verschiedenen Aufgaben darauf, sich ausreichend auszutauschen. Im zweiten Teil des Praktikums müssen sich beide Teams zusammensetzen und jeweils ihre Implementierung dem anderen Team nahebringen!

1.3.2 Termine

Neben den beiden Einführungsterminen gibt es eine Zwischenpräsentation und eine Abschlusspräsentation. Die Zwischenpräsentation findet in der Mitte des Semesters statt. Zu dem Zeitpunkt sollten Sie den ersten Aufgabenteil abgeschlossen haben. Sinn der Zwischenpräsentation ist es, den erreichten Stand der Entwicklung vorzustellen und einen Ausblick auf die verbleibende Arbeit zu geben. Die Dauer des Vortrages sollte 30 Minuten nicht überschreiten.
Die Abschlusspräsentation findet am Ende des Semesters statt. Hier soll Erreichtes und Erlerntes übersichtlich zusammengefasst werden. Dabei steht eine detaillierte Bewertung und Übersicht der Gegenmaßnahmen im Mittelpunkt. Es sollten ebenfalls 30 Minuten für den Vortrag ausreichen.

1.3.3 Zeiteinteilung

Die Durchführung des Praktikums erfolgt in Eigenregie. Sie sollen das Projekt eigentständig erarbeiten. Es bleibt Ihnen frei, wann und wo Sie dies tun. Der Praktikumsraum ist Montag bis Freitag von 7:00 bis 21:00 zugänglich. Da auch andere Veranstaltungen in dem Raum abgehalten werden, nehmen Sie bitte Rücksicht aufeinander. In der Regel ist der Raum aber frei verfügbar.

1.3.4 Tutorzeiten

Zur Betreuung des Praktikums gibt es einen Tutor, der bei Fragen und Problemen zur Seite steht. Der Tutor ist im Sommersemester 2013 zu folgenden Zeiten Anwesend:

- Dienstag 9:00 - 18:00
- Donnerstag 13:30 - 17:30

Zu dem besteht die Möglichkeit per Email Kontakt aufzunehmen: ir@mytum.de

Kapitel 2: Erster Aufgabenteil

2.1 Aufgabenstellung - Zielsetzung

Seitenkanalangriffe stellen eine empfindliche Bedrohung für viele Systeme im Securitysegment dar. Gerade dort wo Hardware außerhalb des kontrollierbaren Bereichs gelangt, in dem es beispielsweise an Kunden ausgehändigt wird. Ein eben solches Szenario wird in diesem Praktikum exemplarisch behandelt. Sie erhalten eine SmartCard, auf der ein kryptographischer Algorithmus ausgeführt wird. Den Schlüssel, den die Karte dabei verwendet, sollen Sie herausfinden. Der Kontext in dem Sie sich befinden, ist das eines geschützten Bezahlfernsehens. Wie bei einem echten PayTV System, kann ein verschlüsselter Videostream „nur“ mit der ausgehändigten SmartCard angesehen werden. Ihre Aufgabe im ersten Teil des Praktikums ist, die gegebene Karte zu vervielfältigen.
Ein Server des Lehrstuhls versendet einen Verschlüsselten Videostream. Diesen kann man mit einem Skript starten: *./client tueisec-sigltv 2000x* . Wobei x durch die Gruppennummer zu ersetzen ist. Damit das Video entschlüsselt werden kann, muss gleichzeitig die PayTV Karte im Kartenleser eingesteckt sein.

Das Videomaterial wird beim Server in 16 Kilobyte große Blöcke geteilt und mittels AES verschlüsselt. Ein Block wird ein Chunk genannt. Für jeden Chunk wird ein Schlüssel k_{chunk} generiert. Mit ihm wird der gesamte Chunk verschlüsselt. Der Chunkkey k_{chunk} wechselt somit alle paar Sekunden im Video und ist unvorhersehbar gewählt. Damit der Client das Video betrachten kann, muss er in den Besitz des Chunkkeys kommen. Nur wenn er für jeden Videochunk den passenden Chunkkey k_{chunk} hat, kann er das Video entschlüsseln. Der Chunkkey muss also vom Server zum Clienten übertragen werden. Dies kann nicht im Klartext geschehen, da sonst jeder, der die Übertragung mithört, das Video ansehen könnte. Der Chunkkey k_{chunk} wird wiederrum selber mit einem Schlüssel k_i mit AES 128bit verschlüsselt und übertragen. Der verschlüsselte Chunkkey wird an jeden Chunk angehängt. Der Client erhält immer ein Paket aus verschlüsseltem Schlüssel k_{chunk} und entsprechendem Videochunk. Um den unverschlüsselten k_{chunk} zu erhalten, muss der Client den verschlüsselten k_{chunk} mit k_i entschlüsseln. Eben dieser Schritt geschieht in der Karte. Die Karte kennt den vorher vereinbarten, konstanten Schlüssel k_i. Das empfangende Gerät des Clienten erhält den verschlüsselten Chunkkey, gibt ihn an die Karte und erhält von der Karte den Chunkkey im Klartext. Anschließend dekodiert das clientseitige Gerät das Video. Im Praktikum ist das Empfangsgerät einfach ein Rechner mit Kartenleser.

Ziel im ersten Teil des Praktikums ist, den Schlüssel k_i mittels DPA zu ermitteln und in eine Klonkarte zu integrieren. Die Klonkarte muss dazu das Standard SmartCard-Protokoll unterstützen und eine AES 128 ECB Entschlüsselung berechnen können.

2.2 Team Klonkarte

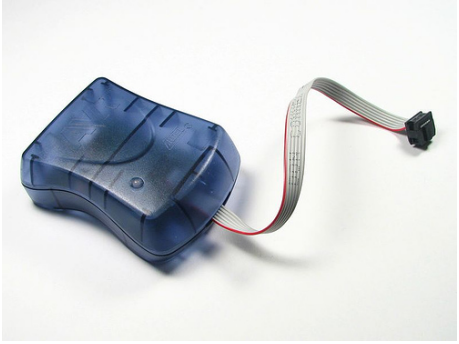


Abbildung 2.1: Programer AVRISPMKII

Abbildung 2.2: SmartCard LabBoard

Sie implementieren das MiniOS für die zweite SmartCard. Auf der zu programmierenden SmartCard ist ein Atmel Atmega 644 Mikroprozessor vorhanden. Richten Sie sich zunächst eine Entwicklungsumgebung Ihrer Wahl ein. Zur Verfügung steht ein AVRISP mkII Programer. Sollten Sie noch keine Erfahrung mit der Programmierung von Atmel Mikroprozessoren haben, empfehlen wir für den Einstieg, ein paar Tutorials zu machen. Solche finden Sie in großer Zahl online. Später wird es hilfreich sein, Debug-Ausgaben über den integrierten UART des Atmega 644 auszugeben. Sehen Sie sich also auch an, wie dieser angesprochen werden kann.
Ziehen Sie zur Implementierung des Systems unbedingt das Datenblatt des Atmega 644 zur Hilfe. Hier finden Sie alle Informationen zur Verwendung der Komponenten des Mikroprozessors, welche Sie benötigen. (Interrupt, Zähler, Ports etc.)

2.2.1 Protokoll - MiniOS

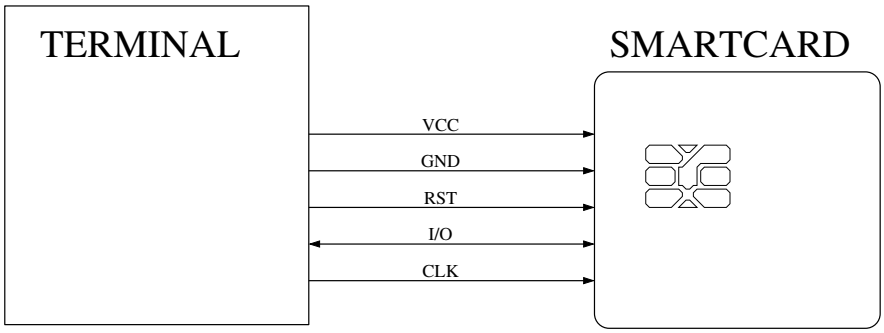


Abbildung 2.3: Grundlegende Konstellation: Terminal, Smartcard

Kontaktbehaftete Chipkarten besitzen im Wesentlichen fünf elektrische Kontakte über die sie betrieben werden (weitere reservierte und proprietäre Kontakte werden nicht verwendet). Über diese Kontakte wird das in der Karte integrierte Mikroprozessorsystem mit Versorgungsspannung (*VCC*), Masse (*GND*) und einem Systemtakt (*CLK*) versorgt. Die beiden übrigen Leitungen

werden als serielle Halbduplex Kommunikationsleitung (*I/O*) und Resetleitung (*RST*) verwendet. Die Kommunikation zwischen Terminal und Smartcard ist bis auf Protokollparameter immer diesselbe und nach *ISO 7816-3* standartisiert. In diesem Standard ist u.a. das T=0 Protokoll beschrieben. Dort finden Sie alle benötigten Details zur Implementierung des Protokolls.

Ihnen steht zur Hilfe ein acht-Kanal-Logicanalyzer von Saleae zur Verfügung. Mit diesem können Sie sehr einfach den Signalverlauf mehrer Leitungen gleichzeitig aufzeichnen und analysieren. Für die Inbetriebnahme müssen Sie die Software auf www.saleae.com herunterladen und starten. Die Webadresse ist auch auf dem Gerät selber vermerkt. Für die I/O-Leitung Sie den Signalanalyzer in der Software verwedenen. Wenn Sie diesen Kanal auf als asynchrone serielle Verbindung konfigurieren, sehen Sie besser welche Bytes übertragen werden.

Bei Inbetriebnahme eines Smartcardsystems legt das Terminal zunächst für eine gewisse Zeit die Versorgungssignale an die Karte an, um dieser eine Initialisierung zu ermöglichen. Nachdem die Karte genug Zeit hatte ihre interne Initialisierung abzuschließen, eröffnet das Terminal mit dem Hochsetzen der Resetleitung die Kommunikation. Es folgt eine Answer to Reset (*ATR*) Sequenz der Karte auf der *I/O*-Leitung. Diese Sequenz enthält kodierte Informationen über die Chipkarte. Diese sind Informationen über die unterstützen Merkmale der Karte und Übertragungsparameter. Die Länge des *ATR* ist variabel. Ihre Aufgabe u.a. ist hier eine minimale ATR-Sequenz zu finden. Details dazu finden Sie im *ISO 7816-3* Standard.
In Abbildung 2.4 ist der Beginn einer solchen Kommunikation gezeigt.

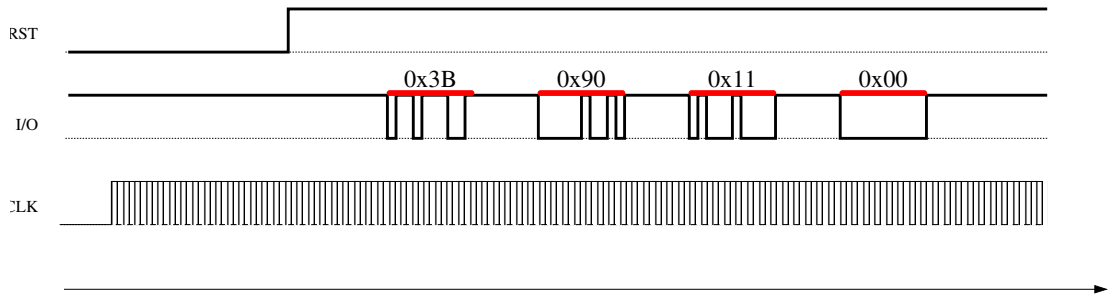


Abbildung 2.4: Übertragung einer Answer-to-Reset Sequenz

Ist der *ATR* erfolgreich abgeschlossen, wartet die Karte auf weitere Instruktionen des Terminals. Ein solcher Befehl besteht zunächst immer aus fünf Bytes, welche Klasse, Befehl und drei Befehlsparameter enthält. Abhängig von dieser Bytesequenz wird eine entsprechende Antwort der Smartcard erwartet. Kommunziert wird in beide Richtungen über die *I/O*-Leitung, dabei sichert das Protokoll stets eindeutig welche Seite an der Reihe ist. Denken Sie auch an den Logicanalyzer, der Ihnen zur Verfügung steht. Verwenden Sie Ihn um die beiden verwendeten Befehle und die gesamte Kommunkation der originalen Karte besser nachzuvollziehen. Dies hilft Ihnen bei der Implementierung der eigenen Karte. Der *ISO 7816-3* Standard enthält zudem eine ausführliche Beschreibung des Aufbaus solcher Befehle.

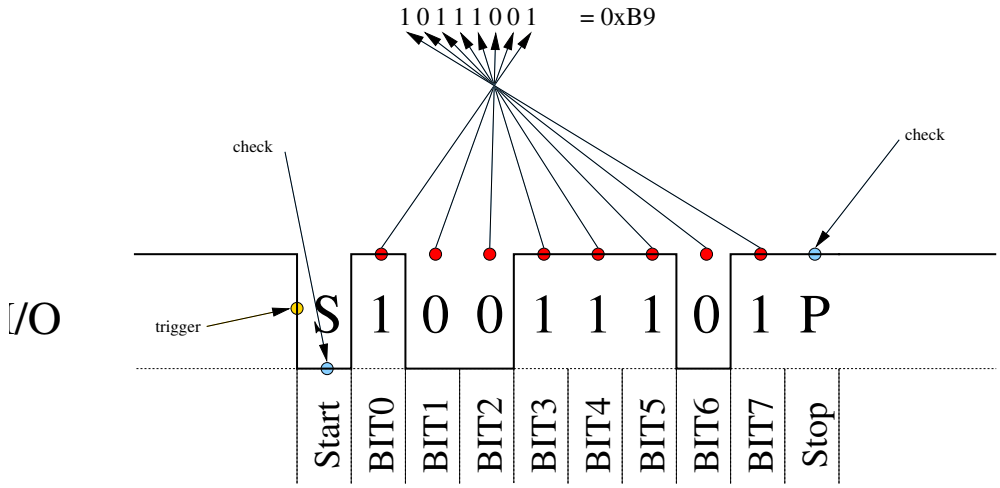


Abbildung 2.5: Ein Byte auf der I/O Leitung

2.2.2 SmartCard UART

Die Übertragung der Daten auf der *I/O*-Leitung ist prinzipiell die selbe wie bei einer üblichen seriellen Schnittstelle (*UART*). Die Kommunikation findet auf unterster Ebene byteweise statt. Empfangs- und Sendeleitung, *RX* und *TX* befinden sich auf einer gemeinsamen Leitung *I/O*. Die Leitung ist im Allgemeinen in einem Tristatezustand mit Pullup, hält also einen Highpegel. Eingeleitet wird die Übertragung eines Bytes stets durch ein Startbit, die *I/O*-Leitung wird für die Dauer einer Elementary-Time-Unit (*ETU*) auf Low gezogen. Es folgen acht Datenbits, daraufhin ein Parity- und ein Stoppbit. Nach dem Stoppbit geht die *I/O*-Leitung wieder in einen Tristatezustand mit Highpegel über. Das Paritätsbit wird so gesetzt, dass die Anzahl der 1 Zustände gerade ist. Jedes Bit wird jeweils für die Dauer einer *ETU* angelegt. Der Signalverlauf ist anhand eines Beispiels in Abbildung 2.5 veranschaulicht.

2.2.3 AES

Das AES Modul stellt die Sicherheitsfunktion der Karte dar. Mithilfe der AES Entschlüsselung wird der Chunkkey k_{chunk} laufend entschlüsselt. Sie benötigen daher lediglich die Entschlüsselungsfunktion, eine Verschlüsselung ist nicht notwendig. Verwendet wird AES 128 bit im ECB Modus. Sie haben die Wahl den Algorithmus selber zu implementieren oder eine bestehende Implementierung zu verwenden. Beachten Sie dabei, dass Sie später Gegenmaßnahmen zur DPA implementieren sollen. Sie müssen sich also in beiden Fällen ausführlich mit der Implementierung auseinandersetzen. Erwägen sie daher, ob eine eigene Implementierung vielleicht nicht einfacher zu durchschauen ist. Wenn Sie eine fremde Implementierung verwenden möchten, kennzeichnen Sie dies in der Dokumentation und nennen Sie die Quelle.

Denken Sie auch daran, während der Berechnung der Entschlüsselung, den Triggerpin auf der Karte zu setzen. Dies benötigen Sie später für Ihre Messungen, wenn Sie Ihren AES angreifen. Außerdem können Sie so die Rechendauer Ihrer Implementierung bestimmen.

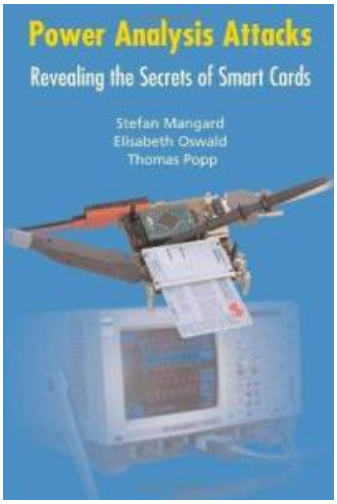
2.3 Team DPA

Sie implementieren die Differential Power Analysis. Diese Aufgabe lösen Sie weitgehend in Matlab. Sollten Sie keine Erfahrung mit Matlab haben, empfehlen wir einige Tutorials zum Einstieg. Diese gibt es auf der offiziellen Mathworks Webseite.

Ziel ist es, den geheimen Schlüssel k_i aus der Karte zu extrahieren. Dies erreichen Sie durch eine geeignete Analyse der Stromverbrauchskurven der Karte. Die Strommessungen erhalten Sie, indem Sie die Karte einige hundert Entschlüsselungen berechnen lassen und dabei den Stromverbrauch und die Ausgangsdaten aufzeichnen. Anschließend werten Sie diese Messdaten aus.

2.3.1 Theorie zur DPA

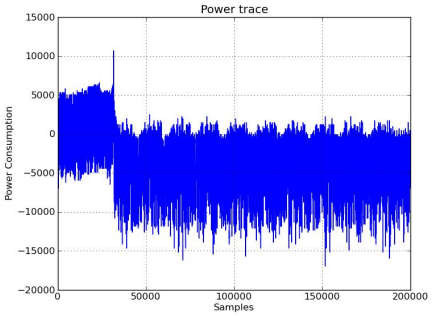
Um den Angriff zu Verstehen, müssen Sie sich zunächst mit dem AES Algorithmus auseinandersetzen. Insbesondere sind die Berechnungsschritte der letzten Runde wichtig.



Eine sehr gute Beschreibung finden Sie in Stefan Mangards Buch *"Power Analysis Attacks - Revealing the Secrets of Smart Cards"*. Dieses können Sie in der TUM Bibliothek ausleihen. Hier sollten Sie alle notwendigen Informationen zum Angriff und Gegenmaßnahmen nachlesen können.

2.3.2 Messung mit Picoscope

Um den Angriff durchzuführen und erfolgreich den Schlüssel k_i zu erlangen, müssen Sie den Stromverbrauch der Karte messen und speichern. Hier messen Sie die Spannung an einem Shuntwiderstand als Maß für den Stromverbrauch des Chips. Auf der gegebenen Karte sind bereits verschiedene solche Shunts vorbereitet. Über das setzten des entsprechenden Jumper können Sie die Größe des Shuntwiderstandes bestimmen. Details hierzu entehmen Sie dem Schaltplan.



Ihr Messinstrument ist das USB Oszilloskop: Picoscope 5204. Damit nehmen Sie Ihre Powertraces auf. Sie können das Picoscope direkt aus Matlab ansprechen. Hierzu ist für Sie bereits eine Matlabfunktion *trace_measurement* vorbereitet. Diese startet die Messung und spricht den Kartenleser an. Der Kartenleser lässt zufällige Daten von der Karte entschlüsseln. Zu jedem Datensatz wird die gemessene Powertrace und der Rückgabewert der AES Entschlüsselung in das angegebene Verzeichnis geschrieben. Sie müssen hierbei lediglich die Zahl Traces, die Samplerate und die Aufnahmedauer vorgeben. Beachten Sie dass die Aufnahmedauer vom Ende der AES Entschlüsselung gemeint ist. Wenn Sie beispielsweise 5 ms für das Samplewindow wählen, werden nur die letzten 5 ms der AES Berechnung gespeichert.
Da der Treiber für das Picoscope leider Fehler enthält, neigt Matlab dazu abzustürzen. Dies ist besonders dann der Fall, wenn große Datenmengen vom Picoscope gehandhabt werden sollen. Achten Sie also darauf die Datenmenge sinnvoll zu reduzieren. Sollte es trotzdem zum Absturz von Matlab kommen, wird die Verbindung zum Picoscope nicht beendet. Um das Picoscope dann zu reseten, müssen Sie es von der Stromversorgung trennen.

Nachfolgend sehen Sie den Code, mit welchem Sie das Picoscope in Ihrem Matlabskript ansprechen können.

```
clear

%% Specify measurement parameters
nTraces      = 250;           % Number of traces to be measured
sampleRate   = 125e6;       % Sampling rate [S/s] (rounded to first sampling rate availa
sampleWindow =  10e-3;      % Sampling temporal window [s]
outDirectory = '/tmp/DPA';  % Output directory where measurements will be saved

%% Start measurement
trace_matrix = trace_measurement(nTraces, sampleRate, sampleWindow, outDirectory);
```

Achten Sie darauf die Messspitzen des Oszilloskop korrekt anzuschließen:

- Stromverbrauch wird über Kanal A gemessen. Hierfür an JP9 anschließen.

- Der Trigger geschieht über den AUXIO auf der Rückseite des Picoscope. An JP5 anschließen.
- Als Masse ist JP8 vorbereitet. Sie können GND aber auch am äußeren Gehäuse der USB-Buchse abgreifen.

2.3.3 Durchführung der DPA

Lesen Sie sich zuallererst in das Thema DPA ein. Sie brauchen ein grundlegendes Verständnis der Idee, welche der Analyse unterliegt.

Beginnen Sie dann damit Ihre Messdaten vorzubereiten. Sprechen Sie das Picoscope an und lesen Sie die Daten ein. Plotten Sie den Stromverlauf, vergleichen Sie mehrere Kurven und versuchen Sie Muster darin zu finden. Definieren Sie geeignete Parameter für die Aufnahmedauer und Traceanzahl.

Beginnen Sie dann die DPA zu Implementieren. Achten Sie auf eine effiziente Programmierung und sinnvolle Datenmengen, um nicht zu viel Rechenzeit zu benötigen. Der DPA Angriff auf ein Schlüsselbyte sollte nicht länger als eine Minute dauern. Optimierte Implementierungen brauchen dafür wenige Sekunden, dies sollte Ihre Referenz sein.

2.3.4 Probleme mit Matlab

Bei der Arbeit mit Matlab und dem Picoscope kommt es häufig zu Abstürzen. Die ist immer dann der Fall wenn große Datenmengen im Spiel sind. Achten daher Sie immer darauf, nicht zu viele Daten auf einmal zu behandeln. Wiederholen Sie lieber öfter Messreihen mit weniger Durchläufen als viele Messdurchläufe auf einmal in Auftrag zu geben. Reihen Sie die Daten dann aneinander sobald sie sicher auf dem Rechner gespeichert sind.

Sollte Matlab beginnen in kurzen Abständen abzustürzen, kann ein Neustart des Systems helfen. Die Problematik stellt sich eigentlich nur im Zusammenhang mit dem Picoscope. Sind die Messdaten einmal aufgenommen und sollen weiter verarbeitet werden, dürfte es nicht mehr zu Abstürzen kommen. Ist dies dennoch der Fall, starten Sie das System neu. Es handelt sich dann warscheinlich noch um Fehler der vorhergegangenen Messungen.

Speichern Sie zwischendurch regelmäßig Ihre Arbeit!

Kapitel 3: Anhang

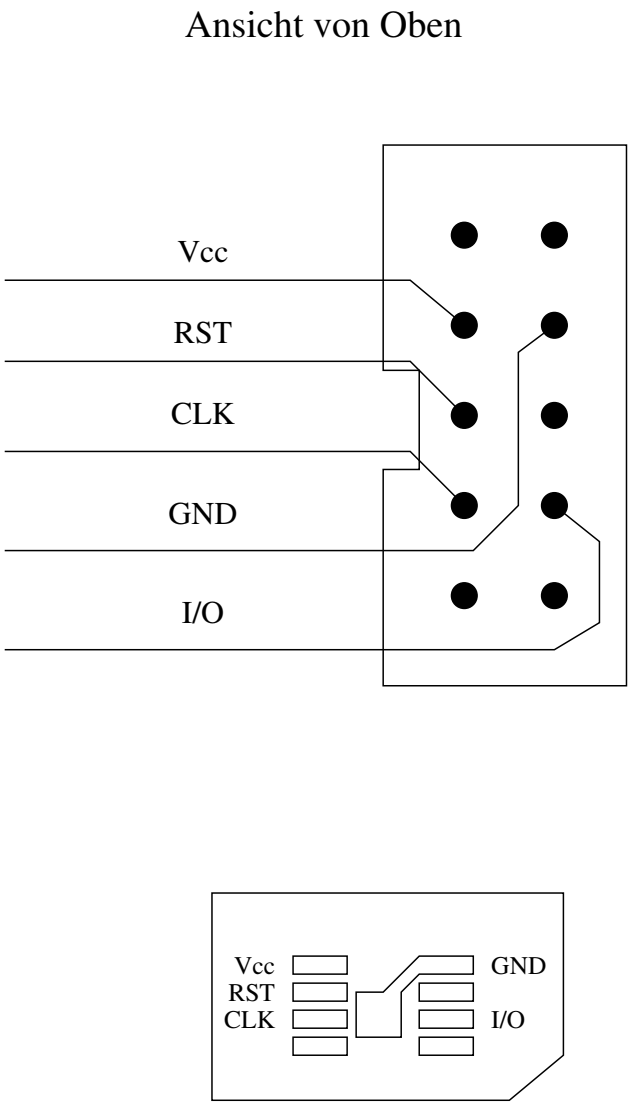


Abbildung 3.1: Kontaktbelegung des SmartCard Interface