

Universität Hamburg
Fachbereich Informatik

Bachelorarbeit

**Seitenkanal-Angriffe auf RSA mithilfe eines
Software Defined Radio**

vorgelegt von

Jan Sönke Ruge
geb. am 18. Februar 1991 in Hamburg
Matrikelnummer 6333676
Studiengang Informatik

eingereicht am 8. Dezember 2015

Betreuer: Ephraim Zimmer
Erstgutachter: Prof. Dr. Hannes Federrath
Zweitgutachter: Dr. Julian Kunkel

Aufgabenstellung

Die Analyse implementierungsspezifischer und physikalischer Charakteristiken der praktischen Realisierung kryptographischer Algorithmen hat das Potential, sogenannte Seitenkanäle aufzudecken. Diese Seitenkanäle transportieren ungewollte und/oder verdeckte Informationen, die sicherheitskritische Parameter oder vertrauliches Material eines ansonsten kryptanalytisch sicheren Algorithmus betreffen. Genkin et al. veröffentlichten im Jahr 2014 Seitenkanal-Angriffe auf die GnuPG-RSA-Implementierung, bei denen durch Messungen des elektrischen Potentials oder elektromagnetischer Signale eines PCs während der RSA-Entschlüsselung der geheime 4096-Bit-Schlüssel extrahiert werden konnte. Mithilfe eines Hochfrequenz-Empfängers (Software Defined Radio, SDR) anstelle eines digitalen Speicheroszilloskops kann der Hardware- und Kostenaufwand bei Seitenkanal-Angriffen möglicherweise reduziert werden. In der vorliegenden Bachelorarbeit soll untersucht werden, inwieweit ein geeigneter SDR-Empfänger (bspw. der USRP B200) zur Durchführung von Seitenkanal-Angriffen auf eine ausgewählte RSA-Implementierung verwendet werden kann.

Abstract

Werden Berechnungen auf einer Hardware ausgeführt, ändert sich der Stromverbrauch der CPU entsprechend den durchgeführten Operationen. Diese Schwankungen können dabei verdeckte Informationen über den Programmablauf beinhalten. Teilweise werden diese Störsignale auch als elektromagnetische Welle abgegeben, welche mit geeigneter Hardware, einem Software-Defined-Radio, empfangen werden kann. Diese Signale können dazu benutzt werden, um Angriffe auf aus mathematischer Sicht, sichere Verschlüsselungsalgorithmen durchzuführen. Hierzu ist eine geeignete Vorverarbeitung der Signale erforderlich. In dieser Arbeit wird untersucht, wie solche Signale aufgearbeitet werden können, um die verdeckten Informationen offen zu legen. Anhand bekannter Angriffe gegen GnuPG wird ein Angriff auf die modulare Potenzierungsmethode von OpenSSL entwickelt und durchgeführt. Dabei wurden 176-Bit des geheimen RSA-CRT Moduls in Folge extrahiert. Diese Arbeit bezieht sich dabei auf aktuelle Forschungsergebnisse [Gen+15], welche kommenden September auf der Konferenz 'Cryptographic Hardware and Embedded Systems 2015' vorgestellt werden.

Inhaltsverzeichnis

1 Einleitung	5
1.1 Ziele und Aufbau	5
1.2 Entstehung von Seitenkanaleffekten	6
1.3 Seitenkanalangriffe auf Laptops	6
1.4 Angriff mittels eines SDR	7
2 Grundlagen RSA	8
2.1 Verfahren	8
2.2 Chinesischer Restsatz / RSA-CRT	9
2.3 Blinding	9
2.4 Binäre Potenzierung	10
2.5 m-Array und Sliding Window Potenzierung	11
3 Grundlagen Digitale Signalverarbeitung	13
3.1 Software Defined Radio	13
3.2 Sampling - Abtastung kontinuierlicher Signale	13
3.3 Spektrogramme(DFT/STFT)	14
3.4 Modulation	15
3.5 DPA - Differential Power Analysis	16
4 Messmethodik und Vorverarbeitung	17
4.1 Antenne	17
4.2 USB Port	18
4.3 Seitenkanaleffekte von HLT	19
4.4 Datenerhebung und Testumgebung	20
4.5 Extraktion der Berechnung	20
4.6 Demodulation	22
4.7 Behandlung von Interrupts	23
5 OpenSSL	26
5.1 Seitenkanaleffekte von Multiplikationen	26
5.2 Laufzeitverhalten und Seitenkanaleffekte	27
5.3 OpenSSL - BN_mod_exp	29
5.4 Angriff auf OpenSSL	30
6 Fazit	33
7 Anhang	35
7.1 Inhalt der DVD	35
7.2 SDR Empfänger	36
7.3 Implementierung	36
7.3.1 DUT	36

7.3.2	Capture	37
7.3.3	Vorverarbeitung	37
7.3.4	Implementierung DPA	38

1 Einleitung

Seitenkanalangriffe sind Angriffe, bei denen die physikalischen Eigenschaften einer Implementierung ausgenutzt werden. Solche Angriffe können dazu genutzt werden, um geheime Schlüssel von einem Gerät zu extrahieren. Das zu testende Gerät wird dabei oft DUT abgekürzt (DUT engl. Device Under Test). Beispielsweise kann die Antwortzeit eines Programms oder der Stromverbrauch eines Prozessors verdeckte Informationen beinhalten [Sta10]. Solche Seitenkanalinformationen können benutzt werden, um interne Zustände oder Änderungen im Programmverlauf zu erkennen. Paul P. Kocher stellte 1998 ein Verfahren vor, mit dem es möglich ist, DES- bzw. RSA-Schlüssel von einfachen Prozessoren zu extrahieren. Dabei wurde der Stromverbrauch einer Smartcard mithilfe eines Oszilloskops digitalisiert und anschließend ein statistisches Verfahren angewendet, um den Schlüssel zu bestimmen [KJJ99]. Diese Art von Angriffen wird Power Analysis Angriff genannt. Aktuelle Ergebnisse aus dem Jahr 2014 zeigen, dass moderne Laptops ebenfalls verwundbar gegen solche Angriffe sind [GPT14].

Um in der Praxis einen Power Analysis Angriff durchzuführen, muss man den Seitenkanal messen können, d.h. sich in unmittelbarer Nähe zum Ziel befinden. Die maximale Reichweite für einen erfolgreichen Angriff ist dabei abhängig von dem verwendeten Seitenkanal. Elektrische Störsignale eines Laptops, welche Seitenkanalinformationen enthalten, können beispielsweise über die Abschirmungen eines Kabels über eine Entfernung von 10m übertragen werden [GPT14]. Teilweise werden diese Seitenkanalinformationen auch als elektromagnetische Welle abgegeben und können mit einer geeigneten Antenne empfangen werden. Dabei wurde eine Reichweite von bis zu 50cm erreicht, wobei kein direkter Kontakt mit dem Zielgerät nötig ist [Gen+15].

Die Angriffe [GST14], [GPT14] und [Gen+15] haben zum Ziel, RSA Schlüssel von einem Laptop zu extrahieren. Da hierbei Eigenschaften der Implementierung von RSA ausgenutzt werden, muss der Angreifer zudem in der Lage sein, eine Entschlüsselung mit dem privaten RSA Schlüssel zu forcieren. Hierbei handelt es sich um einen 'Chosen-Ciphertext-Angriff', bei dem das zu entschlüsselnde Chiffrat frei wählbar sein muss. Hierbei wurde die automatische Signaturüberprüfung von Enigmail ausgenutzt, welche eingehende Emails automatisch mit GnuPG prüft.

1.1 Ziele und Aufbau

In dieser Arbeit wird untersucht, wie die Angriffe auf die RSA Implementierung von GnuPG auf OpenSSL übertragen werden können. Dabei werden die Messmethodik aus [Gen+15] und der Angriff von [GST14] in abgewandelter Form kombiniert, um einen Angriff auf die modulare Potenzierung von OpenSSL durchzuführen. Es werden elektromagnetische Störungen mit einer entsprechenden Antenne gemessen und mithilfe eines digitalen Hochfrequenzempfängers (SDR) gemessen. In Kapitel 2 wird das RSA-Verfahren und dessen Berechnung beschrieben. Die Grundlagen der Digitalen Signalverarbeitung werden in Kapitel 3 erläutert. Anschließend wird in Kapitel 4 beschrieben, mit welcher Methodik die Seitenkanaleffekte gemessen und

aufgearbeitet werden. Diese Erkenntnisse werden anschließend Kapitel 5 verwendet, um einen Angriff gegen die modulare Potenzierungsmethode von RSA durchzuführen.

1.2 Entstehung von Seitenkanaleffekten

Besteht ein Zusammenhang zwischen dem Stromverbrauch der CPU und den durchgeführten Operationen, ist ein elektrischer Seitenkanal vorhanden. Grund hierfür sind die unterschiedlichen Gattergruppen, die je nach Komplexität und Typ unterschiedlich viel Strom verbrauchen. Ein gebräuchliches Modell für den Stromverbrauch einer CMOS Schaltung ist das 'Hamming-Distance Model' [Sta10]. Dieses besagt, dass die Anzahl der Bits, die ihren Zustand ändern, mit dem Stromverbrauch korrelieren.

Interne Busse bilden mit umgebenden Leitern einen Kondensator, welcher in der Lage ist, eine geringe Menge an Energie zu speichern. Ändert ein Bus den Wert von 1 auf 0, bedeutet dies, dass sich das elektrische Potential bspw. von 5V auf 0V ändert. Hierzu muss die gespeicherte Energie abgeleitet werden, was in einem Stromfluß resultiert. Die Menge an Energie ist relativ klein, dieser Prozess wiederholt sich jedoch entsprechend der Taktrate der CPU. Bei einer modernen CPU mit einer Taktrate von 1GHz kann die Entladezeit des Busses maximal 1ns betragen, was in einem relativ hohen Stromfluss resultiert [Sta10].

1.3 Seitenkanalangriffe auf Laptops

In den meisten Fällen richten sich Seitenkanalangriffe gegen eingebettete Systeme wie z.B. Smart Cards [Sta10]. Ein wesentlicher Grund hierfür ist der einfache Aufbau der CPU und die geringe Taktrate von ca. 5MHz. Selbst günstige Oszilloskope bieten eine deutlich höhere zeitliche Auflösung als die Taktrate einer Smart Card. Dies erlaubt, Effekte innerhalb eines CPU-Taktes zu erkennen und auszunutzen. Für moderne Laptops wäre somit eine Abtastrate von mehreren 10 GHz nötig, welche mit günstigen Oszilloskopern nicht erreicht werden kann. Hierfür ist professionelles Equipment erforderlich, das zudem in der Lage sein muss, die anfallenden Daten in Echtzeit auszuwerten oder zwischenspeichern.

Dennoch war es Shamir et al. 2013 gelungen, RSA Private Keys von regulären Laptops zu extrahieren. In diesem Fall wurden akustische Geräusche im Ultraschallbereich (bis zu mehreren hundert kHz), welche von Kondensatoren und Induktivitäten erzeugt werden, als Seitenkanal genutzt. Die relevanten Bauteile waren Teile der Spannungsregulierung der CPU, welche den schwankenden Stromverbrauch der CPU ausgleichen. Obwohl auf einer modernen CPU Instruktionen im GHz Bereich abgearbeitet werden, konnte gezeigt werden, dass dennoch einige Operationen anhand grober Muster über viele Instruktionen unterscheidbar sind. Entscheidend war, dass die relevanten Programmabschnitte mehrfach aufgerufen wurden, sodass eine Verstärkung des Effekts entsteht [GST14].

Ein Jahr später veröffentlichten Genkin et al. einen ähnlichen Angriff [GPT14]. Als Seitenkanal wurden hierbei elektrische Störsignale genutzt, welche ebenfalls von der Spannungsregulierung der CPU verursacht werden. Lediglich in der Art der Messung (akustisch vs. elektrisch) unterscheiden sich diese Angriffe. Als Messpunkte wurde die Abschirmung von I/O Schnittstellen (LAN, USB) genutzt, welche mit der elektrischen Masse des Computers verbunden sind. Diese Schwankungen wurden anschließend analog vorverstärkt, gefiltert und mithilfe eines Digitalen

Speicher Oszilloskops (DSO) digitalisiert. Diese Rohdaten wurden dann weiterverarbeitet, um den eigentlichen Schlüssel zu berechnen. Die Bandbreite der so gemessenen Signale betrug bis zu 2MHz.

1.4 Angriff mittels eines SDR

Eine andere Methode, um hochfrequente elektromagnetische Signale zu digitalisieren, ist ein Software Defined Radio. Dabei handelt es sich um eine spezielle Hardware, die in der Lage ist, ein schmales Frequenzband zu verstärken und bereits analog zu filtern. Sämtliche Konfigurationen und Nachbearbeitung finden dabei durch entsprechende Software statt. Solche Geräte sind bereits ab 20\$ erhältlich (RTL-SDR) [RTLC]

Ein SDR kann somit genutzt werden, um den von [GPT14] vorgestellten Seitenkanal mit wenig Aufwand zu messen. 2015 wurde von derselben Arbeitsgruppe ein entsprechendes Paper veröffentlicht, welches einen solchen Angriff auf GnuPG beschreibt [Gen+15]. Dieses wurde am 27.2.2015 erstmals veröffentlicht und war zum Zeitpunkt der Abgabe des Exposees am 16.2.2015 noch nicht erschienen. Das Resultat ihrer Arbeit war, dass die Störsignale der CPU Spannungsregulierung stark genug sind, um sie aus geringer Distanz (50cm) mit einem SDR und einer entsprechenden Antenne zu messen. Die Arbeit dieses Papers soll im September 2015 im Workshop on 'Cryptographic Hardware and Embedded Systems' (CHES) vorgestellt werden [CHES2015].

:

2 Grundlagen RSA

RSA ist eine von Ronald Rivest, Adi Shamir und Leonard Adleman entwickeltes Public-Key Kryptosystem [RSA78]. Heutzutage findet es eine breite Anwendung von Signaturen bis hin zu Zertifikaten oder Schlüsselaustausch. Im Gegensatz zu symmetrischen Chiffren, welche denselben Key zum Ver- und Entschlüsseln benutzen, verwendet RSA zwei verschiedene: einen öffentlichen und einen privaten Schlüssel. Wird eine Nachricht mit dem öffentlichen Schlüssel verschlüsselt, kann diese nur mit dem dazu passenden privaten Schlüssel entschlüsselt werden. Theoretisch ist es zwar möglich, den privaten Schlüssel aus dem öffentlichen herzuleiten, jedoch ist dies in der Praxis nicht praktikabel (siehe 2.1).

Das Verfahren basiert auf arithmetischen Operationen auf \mathbb{Z}_N , den ganzen Zahlen modulo N . Die Bundesnetzagentur empfiehlt derzeit für N eine Länge von 2048-Bit [Sig].

2.1 Verfahren

Im Folgenden wird das RSA-Verfahren im Detail beschrieben. Als erstes werden zwei zufällige Primzahlen p und q bestimmt und aus diesen der Modul $N = pq$ bestimmt. Mithilfe der beiden Primzahlen kann die Eulersche Phi-Funktion $\varphi(N)$ berechnet werden. Sind p und q nicht bekannt, existiert kein effizientes Verfahren für große Zahlen zur Berechnung von $\varphi(N)$. Der öffentlich bekannte Exponent e wird zufällig gewählt und aus diesem der private Exponent d errechnet. Die Funktion GGT bestimmt hierbei den größten gemeinsamen Teiler zweier Zahlen.

$$\begin{aligned} N &= pq \\ \varphi(N) &= (p-1)(q-1) \\ e &\in \mathbb{Z}_{\varphi(N)} / \{0, 1\} : \text{GGT}(e, \varphi(N)) = 1 \\ d &\equiv e^{-1} \pmod{\varphi(N)} \end{aligned}$$

Der öffentliche Schlüssel besteht aus dem Tupel (N, e) und der private Schlüssel aus (p, q, N, d) . Eine Nachricht m wird mit dem öffentlichen Schlüssel zu dem Geheimtext c verschlüsselt. Dieser Geheimtext kann nur mithilfe des privaten Exponenten d wieder entschlüsselt werden. Die Berechnung geschieht wie folgt:

$$\begin{aligned} c &= m^e \pmod{N} \\ m &= c^d \pmod{N} \end{aligned}$$

Es ist essentiell für die Sicherheit des Verfahrens, dass p und q geheim sind. Zwar ist es theoretisch möglich p und q aus N zu errechnen, jedoch ist dies aufgrund der Länge von N

in der Praxis nicht praktikabel. Dieses Problem ist als Faktorisierungsproblem bekannt. Der größte Modul, der faktorisiert wurde, hatte eine Länge von 768-Bit. Der Rechenaufwand betrug dabei 2000 CPU Jahre auf 2.2GHz-Opteron-CPUs, wobei die Gesamtaufzeit 3 Jahre betrug [Kle+10]. Ist einem Angreifer eine der beiden Primzahlen z. B. p bekannt, ist es ihm möglich die zweite Primzahl $q = N/p$ zu berechnen, was ihm ermöglicht, die Eulersche Phi-Funktion $\varphi(N)$ zu bestimmen. Dadurch ist der Angreifer in der Lage, aus dem öffentlichen Schlüssel den privaten Exponenten $d \equiv e^{-1} \pmod{\varphi(N)}$ zu errechnen [For08]. Dabei ist die Kenntnis der oberen Hälfte einer der Primzahlen bereits ausreichend, um den privaten Schlüssel zu berechnen [Cop97].

2.2 Chinesischer Restsatz / RSA-CRT

In der Praxis wird bei der Verwendung des privaten Schlüssels ein optimiertes Verfahren mit dem Chinesischen Restsatz genutzt. Hierfür ist jedoch die Kenntnis von p und q erforderlich, sodass diese Optimierung bei der Verwendung des öffentlichen Schlüssel nicht zum Einsatz kommen kann.

Nach der RSA Schlüsselgeneration wird der private Schlüssel um folgende Variablen erweitert [SV93].

$$\begin{aligned} d_p &= d \pmod{p-1} \\ d_q &= d \pmod{q-1} \\ q_{inv} &= q^{-1} \pmod{p} \end{aligned}$$

Die Potenzierung mit voller Bitlänge wird durch zwei Potenzierungen und anschließender Rekombination ersetzt. Da p und q etwa die halbe Bitlänge von N besitzen, kann durch Anwendung dieses Verfahrens eine Laufzeitreduzierung um den Faktor 4 erreicht werden [SF08]. Hierbei ist zu beachten, dass der Modul der Potenzierungen jeweils die geheimen Primzahlen p und q sind [SV93].

$$\begin{aligned} c_p &= c^{d_p} \pmod{p} \\ c_q &= c^{d_q} \pmod{q} \\ m &= (q_{inv}(c_p - c_q) \pmod{p})q + c_q \end{aligned}$$

2.3 Blinding

Um RSA Implementierungen vor Seitenkanalangriffen zu schützen ist es möglich, den Geheimtext zu randomisieren. Diese Technik wird Blinding oder Ciphertext Randomization genannt [GST14]. Hierzu wird eine zu N teilerfremde zufällige Zahl r gewählt und der maskierte Geheimtext c_b gebildet. Hierbei ist e der öffentlich bekannte Exponent.

$$c_b = c \cdot r^d \mod N$$

Anschließend wird auf diesem randomisierten Geheimtext die Die RSA Operation durchgeführt.

$$\begin{aligned} m_b &= c_b^e \mod N \\ &= c^e \cdot r^{ed} \mod N \\ &= c^e \cdot r \mod N \end{aligned}$$

Anschließend kann die originale Nachricht wieder zurückgewonnen werden. Hierfür ist das Inverse zu r nötig.

$$\begin{aligned} m &= m_b \cdot r^{-1} \mod N \\ &= c^e \cdot r \cdot r^{-1} \mod N \\ &= c^e \mod N \\ &= m \end{aligned}$$

Da dem Angreifer r nicht bekannt ist, ist es ihm nicht möglich, den Geheimtext der RSA Operation zu kontrollieren. Wie in GnuPG besitzt OpenSSL die Möglichkeit Blinding für RSA einzusetzen. In OpenSSL ist dies seit Version 0.9.7b standardgemäß der Fall (siehe OpenSSL Quelltext crypto/rsa/rsa.h Zeile 205). Um dennoch einen Angriff durchführen zu können, muss diese Option deaktiviert werden. Diese Vorgehensweise wurde ebenfalls von [Gen+15] verwendet.

2.4 Binäre Potenzierung

Ein effizienter Algorithmus zur Berechnung von natürlichen Potenzen ist die binäre Potenzierung [Cor09a]. Dabei werden die Bits des Exponenten vom MSB zum LSB betrachtet und abhängig von dem Wert eines einzelndes Bits unterschiedliche Operationen durchgeführt. Der Pseudocode für eine binäre Potenzierung ist in 2.1 dargestellt. Dabei ist N der Modul, d der Exponent und c die Basis der Potenzierung. Im Fall einer Entschlüsselung mit dem privaten Schlüssel ist c das Chiffrat und kann vom Angreifer frei gewählt werden. In Zeile 5 (Listing 2.1) ist zu erkennen, dass für jedes durchlaufende Bit das Resultat quadriert wird. Ist das i -te Bit $d[i]$ eine 1, wird eine zusätzliche Multiplikation mit c ausgeführt. Nach jeder Operation wird dabei eine Reduktion des Zwischenergebnisses durchgeführt. Dadurch wird das Auftreten von sehr großen Zahlen vermieden, was das Laufzeitverhalten des Verfahrens verbessert. Die Reihenfolge von Multiplikationen und Quadrierungen ist dabei ausschließlich abhängig vom Exponenten.

```

1 //Berechnung von c^d mod N
2 function bin_exp(c, d, N)
3     res = 1
4     c = c mod N
5     for i = n_bits(N)..0
6         res = res^2 mod N
7         if d[i] == 1

```

```

8     res = (res * c) mod N
9
10    return m

```

Listing 2.1: Pseudocode der Binären Potenzierung zur Berechnung natürlicher Potenzen

Die Potenz $c^{10} = c^{1010_2}$ wird beispielsweise durch folgende Sequenz von Quadrierungen(Q) und Multiplikationen(M) berechnet.

$$1 \xrightarrow{Q} 1 \xrightarrow{M} c \xrightarrow{Q} c^2 \xrightarrow{Q} c^4 \xrightarrow{M} c^5 \xrightarrow{Q} c^{10} \quad (2.1)$$

2.5 m-Array und Sliding Window Potenzierung

Eine optimierte Variante der Binären Potenzierung ist die m-Array Methode, welche erstmals von Donald E. Knuth vorgestellt wurde [Knu68]. Dabei werden m Bit des Exponenten zusammen betrachtet, wodurch die Anzahl der Multiplikationen verringert werden kann. Im Fall von $m=1$ entspricht dieser Algorithmus der binären Potenzierung. Listing 2.2 zeigt den Pseudocode für eine m-Array Potenzierung. Zuerst werden die ersten $2^m - 1$ Potenzen von c vorberechnet und in dem Array `pow` gespeichert. Anschließend wird der Exponent d in m -Bit Wörter zerlegt und in `D` gespeichert. Ist die Länge von d nicht durch m teilbar, wird d mit führenden Nullen aufgefüllt, bis eine Zerlegung in m -Bit Wörter möglich ist.

```

1 //Berechnung von  $c^d \bmod N$ 
2 function m_array_exp(c, d, N)
3     c = c mod N
4
5     //pow[i] =  $c^i \bmod N$ 
6     pow[0] = 1
7     for i = 1...m
8         pow[i] = pow[i-1] * c mod N
9
10    D = Zerlegung von d in m-Bit Wörter
11    k = length(D)
12
13    res = D[k-1]
14    for i = k-2...0
15        res = res ^ ( $2^m$ ) mod N // Schritt a
16        if D[i] > 0
17            res = res * pow[D[i]] mod N // Schritt b
18
19    return res

```

Listing 2.2: m-Array Methode zur Potenzierung.

Im Folgenden werden die Zwischenschritte für die Potenzierung von $c^{1898} = c^{11101101010_2}$ für $m=4$ gezeigt. Die Potenz 2^m in Schritt a beträgt dabei $2^4 = 16$. Die Zerlegung von d ist dabei $D = [0111_2, 0110_2, 1010_2] = [7, 6, 10]$. Wie in der Binärrepräsentation des Exponenten zu erkennen ist, agiert Schritt a als bitweise Verschiebung um m Bit auf den Exponenten. In Schritt (b) werden dann die nächsten m Bits aus `D` auf den Exponenten übertragen.

Initialisierung:	res_0	$= c^7$	$= c^{111_2}$
1. Iteration (a):	$res_1 = res_0^{16} = c^{7 \cdot 16} = c^{112}$	$= c^{112}$	$= c^{1110000_2}$
(b):	$res_2 = res_1 \cdot c^6 = c^{112+6} = c^{118}$	$= c^{118}$	$= c^{1110110_2}$
2. Iteration (a):	$res_3 = res_2^{16} = c^{118 \cdot 16} = c^{1888}$	$= c^{1888}$	$= c^{11101100000_2}$
(b):	$res_4 = res_3 \cdot c^{10} = c^{1888+10} = c^{1898}$	$= c^{1898}$	$= c^{11101101010_2}$

Eine Variante der m-Array Methode, ist die Sliding Window Potenzierung. Ist das m -Bit Wort $D[i]$ Null, kann die Multiplikation mit $2^0 = 1$ übersprungen werden. Bei geeigneter Zerlegung des Exponenten in die Wörter $D[i]$, kann die Wahrscheinlichkeit, dass dieser Fall eintritt, erhöht werden. Dabei wird die Anzahl der Multiplikationen im Vergleich zur m-Array Methode reduziert um bis zu 8% [Koç95].

3 Grundlagen Digitale Signalverarbeitung

Digitale Signalverarbeitung beschreibt die Verarbeitung von zeitlich abhängigen Funktionen auf digitalen Systemen. Anwendung finden diese vor allem im Zusammenhang mit gemessenen physikalischen Größen wie Licht (Bilder/Video), Druck (Schall/Audio) und elektrischen Feldern (Radio/Funk). In diesen Fällen handelt es sich um Signale mit komplexen oder reellen Zahlen der Form $s(t) \in \mathbb{R}$ oder $s(t) \in \mathbb{C}$. Um diese Signale digital zu verarbeiten, müssen diese sowohl zeitlich als auch im Wertebereich diskretisiert werden. In diesem Kapitel wird die Gewinnung und Darstellung von Signalen beschrieben. Des Weiteren wird ein Verfahren vorgestellt, mit dem Seitenkanalinformationen aus Rohdaten gewonnen werden können.

3.1 Software Defined Radio

SDR steht für Software Defined Radio und beschreibt eine Technologie, mit der Radiosignale wie z.B. UKW und GSM, analog empfangen und digital verarbeitet werden können. Vorteil dieses Verfahrens ist, dass mit einem Gerät eine Vielzahl von Signalen empfangen oder gesendet werden kann. Lediglich das Programm zur Weiterverarbeitung muss dabei angepasst werden. Ein Frequenzband, ein Teil des elektromagnetischen Spektrums, wird dabei analog gefiltert, verstärkt und anschließend digitalisiert. Alle dafür erforderlichen Komponenten sind dabei in einem SDR Empfänger vorhanden, jedoch ist der nutzbare Frequenzbereich begrenzt. Müssen sehr niedrige Frequenzen gemessen werden, ist ein Upconverter nötig. Dieser verschiebt das zu messende Signal in einen für das SDR nutzbaren Frequenzband.

3.2 Sampling - Abtastung kontinuierlicher Signale

Da auf einer Von-Neumann-Architektur nur diskrete Werte berechnet werden können, werden Signale in einen endlichen Datentyp wie z.B. Float umgewandelt. Dabei entstehen in Abhängigkeit von der Auflösung des Datentyps Rundungsfehler, die das Originalsignal verfälschen. Der Grad dieser Verfälschung ist abhängig von der Auflösung des verwendeten Digital Analog Konverters (ADC). Die Auflösung bei SDR Empfängern beträgt dabei idR. 8bit (z.B. RTL-SDR) [RTLb] bis 12-14 Bit (USRP) [Res10].

Entsprechende Diskretisierungsfehler treten auch in der zeitlichen Domäne auf. Das Signal wird dabei mit einer bestimmten Frequenz abgetastet, auch bekannt als Abtastrate (engl. sampling rate oder sampling frequency). Diese wird üblicherweise in Hertz (Hz) bzw. Samples pro Sekunde (SPS) angegeben ($1\text{MSPS} = 1\text{MHz} = 10^6$ Messungen pro Sekunde). Dabei wird das zuvor zeitlich kontinuierliche Signal auf diskrete Zeitschritte heruntergebrochen.

Handelt es sich um reelle Samples, ist die höchste darstellbare Frequenz 2 Samples pro Periode. Dabei werden Wellenberg und -tal jeweils durch ein Sample repräsentiert. Enthält das Signal dabei Frequenzanteile höher als die halbe Abtastrate, kann es hierbei ebenfalls zu Verfälschungen des Signals kommen. Aus diesem Grund muss das Signal vor der Digitalisierung in seiner

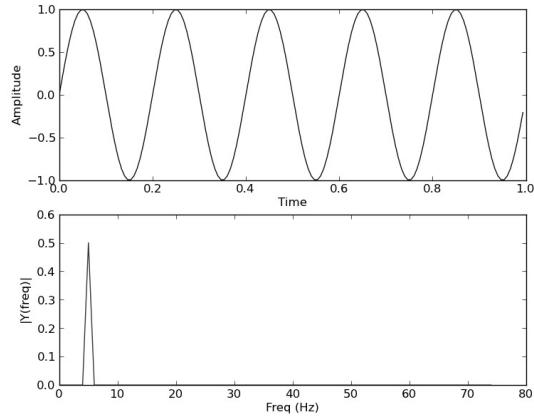


Abbildung 3.1: Der ober Graph zeigt ein Beispieldesign bestehend aus einer Sinusschwingung. Der untere Graph zeigt die dazugehörige DFT.

Bandbreite begrenzt werden. Moderne SDR Empfänger benutzen hierfür Tiefpassfilter vor dem eigentlichen Analog-Digital-Wandler. Dieser Effekt ist als das Nyquist-Shannon-Abtasttheorem bekannt [Ste].

Handelt es sich um ein komplexes Signal, wird jeder Sample durch einen Reel- und einen Imaginärteil dargestellt. Dadurch lassen sich ebenfalls negative Frequenzen darstellen, so dass in diesem Fall die Bandbreite der Abtastrate des Signals entspricht.

3.3 Spektrogramme(DFT/STFT)

Ein Spektrogramm ist eine Darstellung von Signalen und ist auch unter den Namen Waterfall Diagramm oder STFT (Short-time Fourier Transform) bekannt. Die Berechnung basiert auf der Diskreten Fourier Transformation (DFT), welche ein wichtiges Verfahren aus der Signalttheorie ist. Die Laufzeit dieses Verfahrens liegt dabei in $O(n^2)$. Mithilfe der DFT ist es möglich, ein Signal von der Zeitdomäne in die Frequenzdomäne umzuwandeln. Der meist verwendete Algorithmus zur Berechnung einer DFT ist FFT (Fast Fourier Transformation) mit einer Laufzeit von $O(n \cdot \log(n))$. Dabei wird ein Fenster von mehreren Samples (idR. 2^n) in ein Frequenzspektrum gleicher Länge umgewandelt. Die einzelnen Elemente der DFT werden Bins genannt, welche u.a. Informationen über die Stärke der entsprechenden Frequenz enthalten [Cor09b]. Abbildung 3.1 zeigt ein Beispieldesign und die dazugehörige DFT. Es ist ersichtlich, dass das Signal aus nur einer Frequenz von ca. 5Hz besteht.

Eine DFT kann jedoch nur über einer begrenzten Eingabelänge, jedoch nicht über unendliche Datenströme berechnet werden. Um Abhilfe zu schaffen, wird die DFT nur auf ein kurzes Zeitfenster des Signals angewendet. Dieses Fenster wird dann mit einer bestimmten Schrittweite über das Signal verschoben. Bei jedem Schritt wird eine neue DFT von dem aktuellen Zeitfenster berechnet. Das Resultat einer STFT ist eine 2D-Matrix mit jeweils einer Zeit- und einer Frequenzachse [Sel05]. Diese kann dazu benutzt werden, die Intensität einer bestimmten Frequenz zu einem beliebigen Zeitpunkt im Signal zu bestimmen.

Die Länge des Fensters stellt dabei einen Kompromiss aus Zeit- und Frequenzauflösung dar. Je breiter das Fenster, desto mehr Bins stehen zur Unterscheidung der Frequenz zur Verfügung.

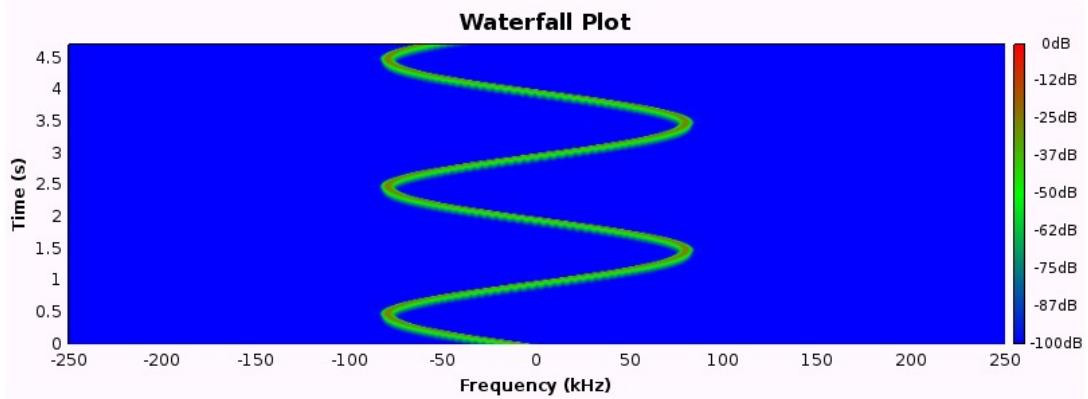


Abbildung 3.2: Spektrogramm einer frequenzmodulierten 0.5 Hz Sinusschwingung. In dieser Darstellung ist das Nutzsignal zu erkennen.

Dies führt zu einer hohen Frequenzauflösung, jedoch kann das zeitliche Auftreten der Frequenz innerhalb des Fensters nicht weiter eingegrenzt werden. Dies führt somit zu einer schlechteren zeitlichen Auflösung.

3.4 Modulation

Nutzsignale wie z.B. Audio können nicht direkt über Funk übertragen werden, hierzu ist es nötig, das Signal zu modulieren. Die Rückgewinnung des Nutz- aus dem Rohsignal wird Demodulation genannt. Zudem ist es erforderlich das Signal zu filtern, um Störsignale und Rauschen zu entfernen. Details zur Implementierung der Vorverarbeitung, wie sie von [Gen+15] genutzt wurde, sind jedoch nicht bekannt.

Die einfachste Form der Modulation ist die Amplitudenmodulation, welche das Nutzsignal über die Intensität einer Trägerfrequenz codiert. Ein Verfahren zum Modulieren von analogen Signalen ist die Frequenzmodulation, welche beispielsweise Anwendung beim UKW Rundfunk findet. Hierzu wird ein Trägersignal mit einer bestimmten Frequenz erzeugt. Das Nutzsignal wird dabei durch eine Frequenzänderung des Trägersignals moduliert, wobei die Intensität des Trägersignals konstant bleibt. Abbildung 3.2 zeigt das Spektrogramm einer frequenzmodulierten 0.5Hz Sinus Schwingung. Das Nutzsignal ist in dieser Darstellung erkennbar. Genkin et. al. haben bei ihren Versuchen festgestellt, dass Seitenkanalinformationen frequenzmoduliert übertragen werden [Gen+15].

In dieser Arbeit wird zur Vorverarbeitung und Demodulation das GNU Radio Framework benutzt [RTLa]. Dieses bietet einen großen Funktionsumfang aus der Signalverarbeitung, wie z.B. digitale Filter und verschiedene Demodulationen. Zudem ist dieses Framework kompatibel mit den meisten SDR Empfängern. Dabei werden einzelne Operationen als Blöcke abgebildet, welche in C++ implementiert sind. Diese können über eine GUI zu einem Flowgraph verbunden werden. Ein Flowgraph wird anschließend in Python code übersetzt (Dateiname: top_block.py) und ausgeführt. Diese kann wiederum in jede Python Applikation eingebunden werden.

3.5 DPA - Differential Power Analysis

Differential Power Analysis (kurz DPA) ist eine Technik, mit der Seitenkanalinformationen aus Rohdaten gewonnen werden können. Dieses Verfahren wurde erstmals von Paul Kocher am Beispiel von DES vorgestellt [KJJ99].

Ziel einer DPA ist zu bestimmen, ob zwei verschiedene Programmeingaben, zu unterschiedlichen Seitenkanaleffekten führen. Zuerst werden zwei geeignete Argumente a und b gewählt, welche zu unterschiedlichen Berechnungen führen könnten. Anschließend führt man wiederholt Messungen in zufälliger Reihenfolge durch. Diese Spuren werden anschließend anhand der Eingabe in zwei disjunkte Gruppen A und B unterteilt. Sei $E(A)$ der Mittelwert der Messungen mit dem Argument a , dann können für die Differenz $E(A) - E(B)$ zwei Fälle eintreten:

- $|E(A) - E(B)| = 0$ für $n \rightarrow \infty$: Die Messungen der Gruppe A unterscheiden sich nicht von den Messungen der Gruppe B . In diesem Fall hat die Wahl des Arguments aus $\{a, b\}$ keinen Einfluss auf die Messung, denn es ist kein Unterschied feststellbar. Dieser Fall tritt ebenfalls ein, wenn die Zuordnung der Messungen in die Gruppen A, B zufällig gewählt wurden.
- $|E(A) - E(B)| > 0$ für $n \rightarrow \infty$: Die Wahl der Eingabe aus $\{a, b\}$ hat einen Einfluss auf die Messung, demnach führen a und b zu unterschiedlichen Seitenkanaleffekten. Ist das zugrundeliegende Programm bekannt können somit Rückschlüsse auf den Programmverlauf gezogen werden.

Dieses Verfahren lässt sich durch Verwendung von Matrizen auch auf mehrdimensionale Daten wie Zeitserien oder Spektrogramme (2D Matrix) anwenden. Handelt es sich dabei um zeitabhängige Signale, ist das Ergebnis einer DPA Analyse zudem abhängig von der Qualität der Ausrichtung. Eine Grundannahme ist, dass alle Operationen innerhalb der Messung ausgerichtet sind, d.h. zum selben Zeitpunkt stattfinden. Dadurch werden sich Seitenkanaleffekte über mehrere Messungen hinweg verstärken. Ist dies nicht der Fall, verteilen sich Seitenkanaleffekte über mehrere Samples, wodurch diese abgeschwächt werden.

Wie aus Abbildung 4.9 ersichtlich setzen sich die gemessenen Spuren aus mehreren Frequenzanteilen zusammen zusammen. Wird eine DPA Analyse auf Sinusschwingungen durchgeführt, ist es essentiell, dass die einzelnen Spuren entsprechend ihrer Phase ausgerichtet sind. Ansonsten würden sich positive und negative Halbwellen gegeneinander eliminieren, sodass $E(A) = 0$ für $n \rightarrow \infty$. Besteht das Signal aus mehreren Frequenzanteilen, ist eine solche Ausrichtung im Allgemeinen nicht möglich. Eine Lösung für dieses Problem ist die DPA Analyse auf Spektrogramme der Signale anzuwenden, da in diesen lediglich die Intensität der Sinuschwingungen enthalten ist. Das Resultat ist ein differenzielles Spektrogramm, aus welchem ersichtlich ist, wann sich zwei Ausführungen und in welchen Frequenzbereichen unterscheiden [Sch+10].

4 Messmethodik und Vorverarbeitung

In diesem Kapitel wird beschrieben, mit welchen Methoden die Messungen durchgeführt wurden. Die elektromagnetische Strahlung wurde dabei mit einer entsprechenden Antenne gemessen und mithilfe eines SDR digitalisiert. Dieses Verfahren bietet die Möglichkeit, den Angriff ohne physischen Kontakt durchzuführen. Die Art und das Vorhandensein von Seitenkanaleffekten ist zudem abhängig von der Hardware des DUT. Einige Modelle zeigen kaum oder gar keine Seitenkanaleffekte.

Um die verdeckten Informationen zu extrahieren, müssen die Spuren erst aufbereitet werden. Zudem ist es nötig, die Ausführung des Programms und die Aufnahme der Spuren zu automatisieren. Da einzelne Spuren zu viel Rauschen enthalten, ist es nötig, einen Durchschnittswert zu bilden. Hierzu ist es notwendig, den Zeitabschnitt der Berechnung innerhalb der Spur zu bestimmen und zu extrahieren. Dann können die Spuren demoduliert und einer DPA Analyse unterzogen werden.

4.1 Antenne

Hochfrequente Wechselströme können nicht nur über elektrische Leiter übertragen werden, sondern werden auch teilweise in elektromagnetische Wellen umgewandelt. Dabei agiert der stromführende Leiter als Sendeantenne, der einen Teil der Energie als elektromagnetische Welle abstrahlt. Diese Wellen können auch durch den leeren Raum propagieren und die meisten nicht leitenden Materialien wie z.B. Kunststoffe, Luft, Holz durchdringen.

Diese elektromagnetischen Wellen können über eine Antenne aufgefangen und in ein elektrisches Signal umgewandelt werden. Ein SDR Empfänger ist empfindlich genug, um dieses Signal zu digitalisieren.

Eine mögliche Bauweise für eine Antenne wurde von [Gen+15] vorgestellt, mit welcher ein vollständiger Angriff durchgeführt wurde. Diese Bauart wird Magnetfeldsonde genannt und ist wegen ihrer Spulenform besonders empfindlich für wechselnde Magnetfelder [Smi99]. Abbildung 4.1 zeigt eine schematische Darstellung einer Magnetfeldsonde mit einer Windung. Die von [Gen+15] und in dieser Arbeit verwendete Sonde besitzt 3 Windungen. Links ist der Signalausgang, der zum SDR führt (1). Die Sonde selbst besteht im Wesentlichen aus einer Luftpule aus einem Koaxialkabel und einer 'Center-Gap' (2), einer Unterbrechung der Abschirmung in der Mitte des Kabels. Am Ende des Kabels ist der Innenleiter elektrisch leitend mit der Abschirmung des Kabels und des Signalausgangs verbunden (3). Als Ausgangsmaterial wurde ein handelsübliches TV-Antennenkabel verwendet.

Abbildung 4.2 zeigt den in dieser Arbeit verwendeten Versuchsaufbau mit der Antenne. Der Computer (links) ist das DUT, welches vom Laptop (rechts) angegriffen wird. Die beiden Systeme sind lediglich zur Kommunikation über das Netzwerk verbunden. Da die elektromagnetische Strahlung durch das geschlossene Metallgehäuse des DUT abgeschirmt wird, ist es nötig, dieses

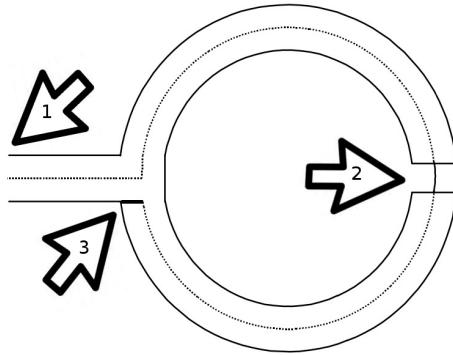


Abbildung 4.1: Schematische Darstellung einer Magnetfeldsonde mit einer Windung aus [Smi99]

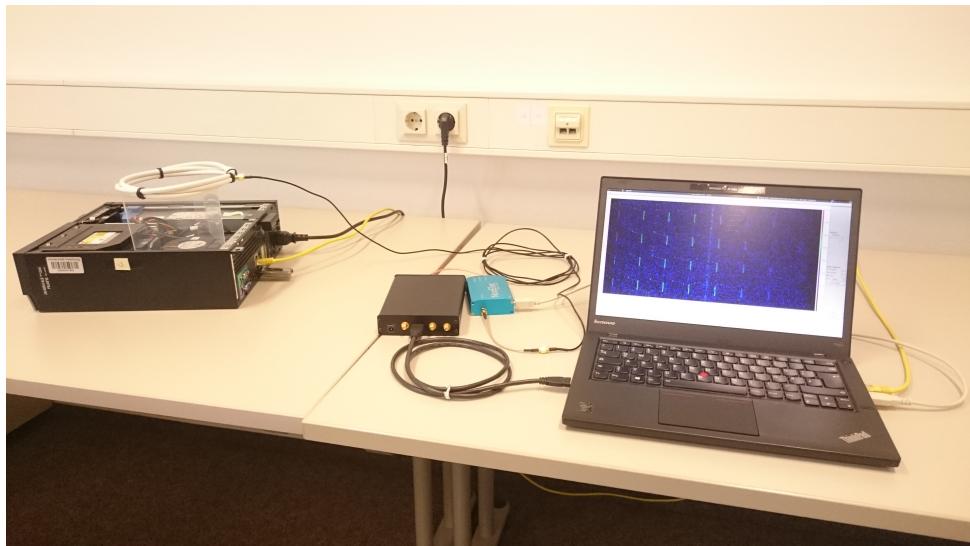


Abbildung 4.2: Dieser Versuchsaufbau mit der Antenne ist der Arbeit von [Gen+15] nachempfunden.

zu öffnen. Der Abstand der Antenne beträgt in diesem Versuchsaufbau 10cm zum CPU-Kühler. In der Mitte ist der SDR-Empfänger (schwarz) und der Upconverter (blau) zu erkennen.

4.2 USB Port

Der USB Port ist eine Schnittstelle, die bei fast allen modernen Laptops vorhanden ist. Da USB Geräte in der Regel über keine eigene Stroversorgung verfügen, wird die Vorsorgungsspannung von dem USB Host gestellt. Die bereitgestellte 5V Gleichspannung besitzt jedoch eine geringe Restwelligkeit, die unter anderem von dem Netzteil und anderen Komponenten in der Umgebung erzeugt werden. Diese Restwelligkeit korreliert dabei ebenfalls mit den auf der CPU durchgeführten Operationen [OS06]. Mit einem geeigneten USB Adapter kann diese gemessen werden, wobei ähnliche Resultate wie mit der Antenne erzielt wurden. Dieses Verfahren bietet jedoch weniger Kallibrierungsmöglichkeiten als wenn eine Antenne verwendet wird. Abbildung 4.3 zeigt den hierfür verwendeten USB Adapter. Dabei wird der Antenneneingang des SDR mit dem +5V Pin des USB Ports verbunden. Da die relativ hohe Gleichspannung unter Umständen

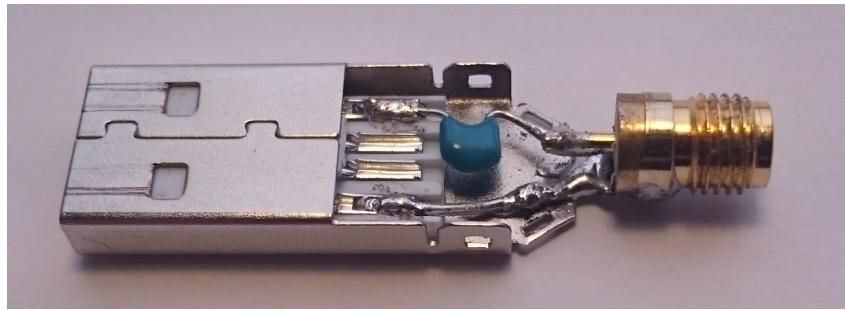


Abbildung 4.3: Mit diesem USB Adapter wurden ähnliche Ergebnisse wie mit der Antenne erzielt.

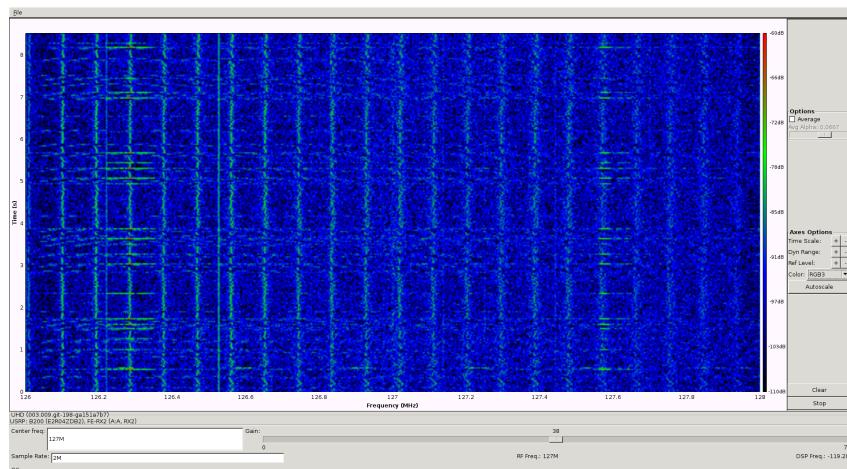


Abbildung 4.4: Dies Bild zeigt das Spektrogramm eines Ubuntu im Leerlauf von 0 Hz - 2 MHz.
Es sind deutlich einige Trägerfrequenzen sichtbar, welche über die Zeit konstant sind.

den Empfänger beschädigen kann, wurde zum Schutz ein 100nF Kondensator in Serie geschaltet. Dieser ist undurchlässig für Gleichspannungen, jedoch transparent für Wechselströme. Die Abschirmung des Antennenkabels wurde mit der Masse und somit mit der Abschirmung des USB Ports verbunden.

4.3 Seitenkanaleffekte von HLT

Seitenkanaleffekte zwischen einer ausgelasteten CPU und einer CPU im Leerlauf (Assembler Instruktion HLT) lassen sich schon mit einfachen Mitteln zeigen. Hierzu wird das Tool uhd_fft mit der Option -W benutzt, welches ein Spektrogramm in Echtzeit darstellt. Parameter wie Frequenz, Bandbreite und Verstärkungsgrad können dabei frei gewählt werden.

```

1 | import time
2 |
3 | while 1:
4 |     for i in xrange(40000000): pass
5 |     time.sleep(1)

```

Listing 4.1: Einfaches Programm zum Erzeugen von Seitenkanaleffekten.

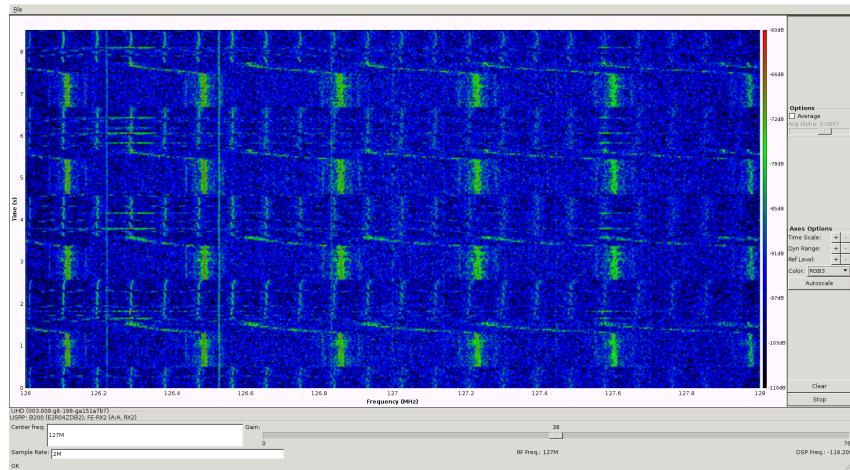


Abbildung 4.5: Dies Bild zeigt das gemessene Spektrogramm während der Ausführung von 4.1.
Es sind deutliche Unterschiede zwischen Berechnung und Leerlauf erkennbar.

Abbildung 4.4 zeigt das Spektrogramm (0 Hz - 2 MHz) des DUT im Leerlauf. Es sind einige Trägerfrequenzen erkennbar (senkrechte Striche) welche von einem Rauschen überlagert sind. Im Wesentlichen ist das Muster über die Zeit jedoch konstant.

Das Python Skript 4.1 wird verwendet, um die CPU des DUT abwechselnd in den ausgelasteten Zustand oder Leerlauf zu versetzen. Es ist erkennbar, dass zeitweise neue Trägerfrequenzen ein regelmäßiges Muster aufweisen, welches durch die beiden verschiedenen Programmabschnitte erzeugt wird. Es lässt sich somit anhand des Spektrogramms entscheiden, ob eine CPU sich im Leerlauf befindet oder nicht. Dieses Verhalten wird im Folgenden genutzt, um den Zeitabschnitt einer Berechnung innerhalb der Spur zu bestimmen.

4.4 Datenerhebung und Testumgebung

Um Seitenkanaleffekte von Programmen zu analysieren ist es notwendig, diese in einer kontrollierten Umgebung zu starten. Zu diesem Zweck wird ein einfacher TCP Server implementiert, der auf Anfragen zum Starten eines Testprogrammes wartet. Diese Anfrage enthält Argumente, die dem Programm beim Start übergeben werden. Nach Beendigung der Ausführung erhält der Client eine Antwort als Bestätigung. Die beiden Nachrichten markieren dabei grob Anfang und Ende der Berechnung und können somit vom Client zur Synchronisierung genutzt werden.

4.5 Extraktion der Berechnung

Um mehrere Spuren miteinander vergleichen zu können ist es notwendig, diese gegeneinander auszurichten, da die Kommunikation über Netzwerk und mit dem SDR zu ungeau ist. Das Ziel dabei ist, innerhalb des Spektrogramms die Ausführung des Prorgramms zu erkennen und den entsprechenden Zeitabschnitt zu extrahieren. Hierzu werden die Erkenntnisse aus der Beobachtung (siehe Seitenkanaleffekte von HLT) genutzt, dass einzelne Trägerfrequenzen ihre Intensität ändern, wenn die CPU ausgelastet ist. Somit kann der Intensitätsverlauf einer Trägerfrequenz als Anhaltspunkt zur Ausrichtung dienen. Da ein Spektrogramm den Intensitätsverlauf aller

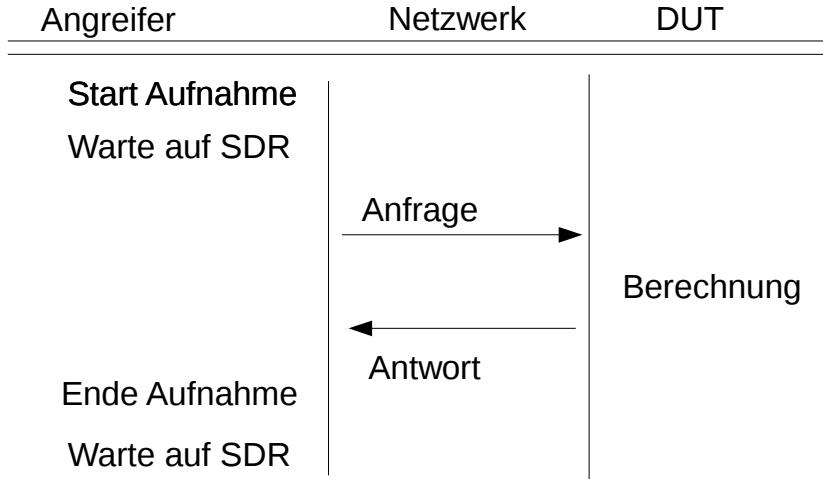


Abbildung 4.6: Kommunikationsdiagramm für die Kommunikation zwischen Angreifer und DUT.

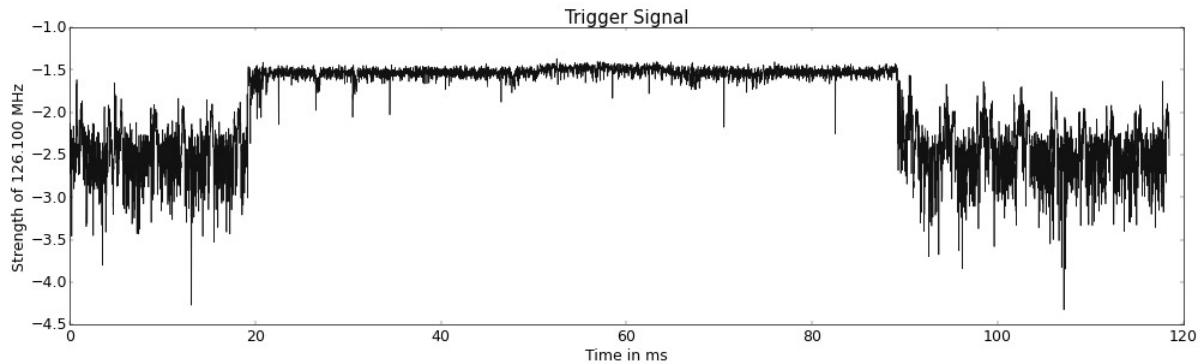


Abbildung 4.7: Der Intensitätsverlauf einer Trägerfrequenz abgeleitet aus dem STFT. Der Zeitraum der Programmausführung ist klar zu erkennen, jedoch ist das Signal durch ein Rauschen überlagert.

Frequenzen enthält, kann der Intensitätsverlauf einer Frequenz aus diesem einfach abgeleitet werden.

Abb. 4.7 zeigt die Intensität einer Trägerfrequenz (y-Achse) über die Zeit (x-Achse). Obwohl der Bereich der Ausführung klar zu erkennen ist, ist das Signal durch Rauschen und mehrere Spitzen teilweise gestört. Dennoch ist der Zeitabschnitt der Ausführung klar zu erkennen. Eine einfache Methode, wäre einen Schwellwert zu definieren, wobei ein Unterschreiten das Ende der Berechnung markiert. Bei großen Spikes kann es jedoch zur Fehlausrichtung anhand der Spitzen kommen. Eine Möglichkeit, dieses Problem zu umgehen, ist die Glättung des Signals. Hierzu wird eine exponentielle Glättung gewählt, welche ohne großen Aufwand zu implementieren ist. Der Grad der Glättung lässt sich dabei über den Faktor $0 < \beta < 1$ einstellen [SS01].

$$stft_{avg}(t, f) = \begin{cases} 0 & t = 0 \\ stft(t, f) \cdot \beta + stft_{avg}(t - 1, f) \cdot (\beta - 1) & \text{sonst} \end{cases}$$

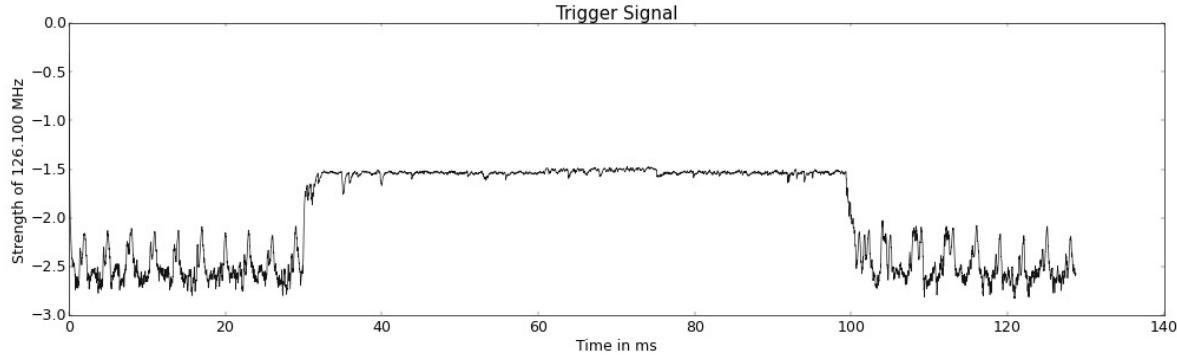


Abbildung 4.8: Geglätteter Intensitätsverlauf einer Trägerfrequenz, wobei $\beta = 0.9$ gewählt. Der Zeitraum der Programmausführung ist klar zu erkennen. Sowohl das Rauschen, als auch die Spitzen werden deutlich abgeschwächt.

Nach Anwendung dieser Funktion mit dem Parameter $\beta = 0.9$ hat das Signal die Form, wie in Abbildung 4.8 zu sehen ist. Die Spitzen werden dabei stark abgeschwächt, sodass ein Schwellwert für eine Ausrichtung möglich ist. Vergleicht man jedoch mehrere Messungen, so ist erkennbar, dass die Laufzeit von mehreren Aufrufen desselben Programms nicht konstant sind. Dies ist u.a. auf den nicht deterministischen Prozessstart zurückzuführen. Aus diesem Grund werden die Spuren am Ende der Ausführung ausgerichtet, da sonst eine genaue Ausrichtung nicht gewährleistet werden kann. Um diesen Zeitpunkt zu bestimmen, wird ein Schwellwert definiert und anschließend das geglättete Signal durchlaufen. Ein Unterschreiten des Schwellwertes markiert dabei das Ende der Ausführung.

4.6 Demodulation

Betrachtet man die rohe Spur während der Ausführung, dann fällt auf, dass die Bandbreite einiger Trägerfrequenzen zeitweise zunimmt. Dies ist durch ein Rauschen in der Frequenzachse zu erkennen. Betrachtet man die Änderung der Trägerfrequenz über die Zeit, spricht man von einer Frequenzmodulation. Dieses Verfahren wurde von Tromer et al. zur Demodulation des Seitenkanals genutzt [Gen+15].

Um dieses Verfahren anzuwenden wird das GNU Radio Framework benutzt und ein entsprechender Flowgraph erstellt. Dabei werden mehrere Filter verwendet, die es erlauben, einzelne Trägerfrequenzen zu extrahieren. Zur Konfiguration des Flowgraphen werden Selektoren benutzt, die es erlauben den Datenfluss anzupassen, um z.B. eine Modulationsart auszuwählen. Diese Parameter können später über eine API geändert werden. Dies bietet den Vorteil, gespeicherte Spuren mit verschiedenen Einstellungen zu testen.

Die Demodulation erfolgt in folgenden Schritten: Der verwendete Flowgraph ist in Abbildung 7.1 zu sehen.

1. 'Frequency Xlating FIR Filter': Dient zur Verschiebung der Spur in der Frequenzdomäne um einen bestimmten Wert. Dies dient zur nachträglichen Selektion von Trägerfrequenzen aus einem Signal. Dazu wird das gewünschte Signal um 0Hz Zentriert.

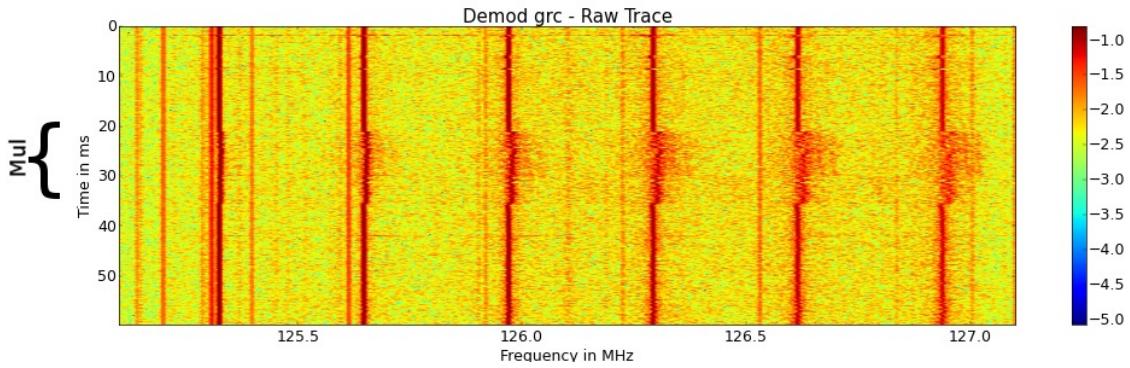


Abbildung 4.9: Spektrogramm der ausgerichteten Spur vor der Demodulation. Es ist zu erkennen, dass zeitweise die Bandbreite einiger Trägerfrequenzen zunimmt. Dies ist charakteristisch für eine Frequenzmodulation, bei der Signale über die Verschiebung einer Trägerfrequenz moduliert werden. Das verwendete Testprogramm findet sich unter 5.1.

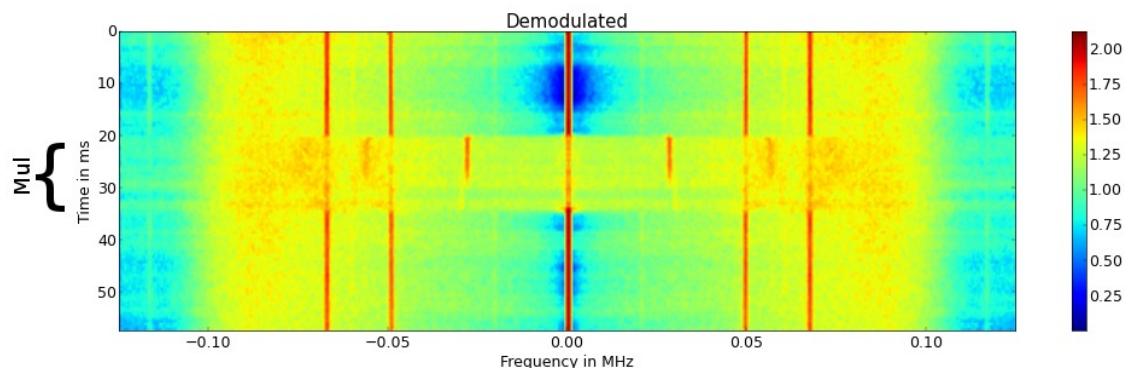


Abbildung 4.10: Mittelwert von 60 Spektrogrammen der demodulierten Spur. Das verwendete Testprogramm findet sich unter 5.1.

2. 'Low Pass Filter': Entfernt alle Frequenzanteile die größer sind als ein angegebener Schwellwert. Dies dient dazu, unerwünschte Frequenzanteile zu entfernen, sodass nur die gewünschte Trägerfrequenz vorhanden ist.
3. Demodulation: Unterstützung für Amplitudenmodulation ('AM Demod') oder Frequenzmodulation ('Quadrature Demod')
4. 'Band Pass Filter': Extrahiert ein Frequenzband aus der demodulierten Spur. Dies ist äquivalent zu einer Serie aus Hoch- und Tiefpassfilter.
5. 'Rational Resampler': Da die Bandbreite des Signals u.U. durch die Tiefpassfilter begrenzt wird, lässt sich die Abtastrate nach dem Nyquist-Shannon-Abtasttheorem reduzieren. Dies führt dazu, dass weitaus weniger Samples in den folgenden Schritten berechnet werden müssen.

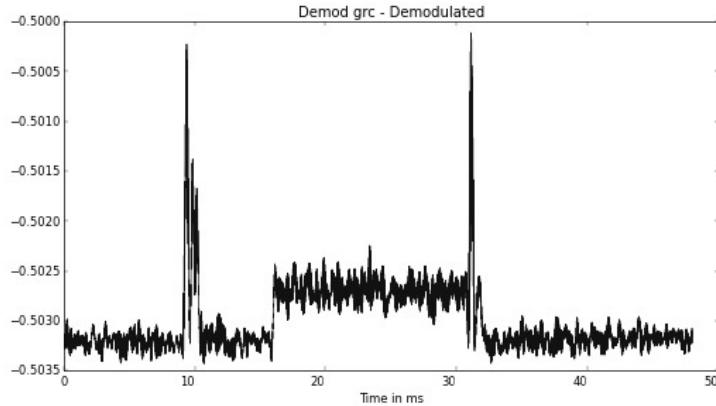


Abbildung 4.11: Demodulierte Spur, dargestellt als Zeitserie, welche durch zwei Interrupts gestört ist.

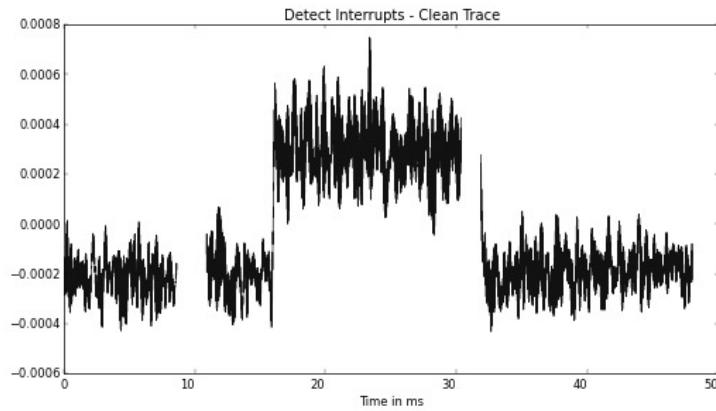


Abbildung 4.12: Interrupts wurden entfernt und durch NaN ersetzt.

4.7 Behandlung von Interrupts

Interrupts sind Unterbrechungen des Programmflusses und werden entweder durch Software- bzw. Hardware-Ereignisse ausgelöst. Dabei wird der aktuelle Programmfluss unterbrochen und eine Interrupt Service Routine aufgerufen. Anschließend wird der alte Programmfluss wieder aufgenommen. Solche Interrupts erzeugen kurze starke Störungen innerhalb der Spur. Durch die sehr große Varianz entstehen Störungen beim Bilden des Mittelwertes, wodurch der eigentliche Seitenkanal überdeckt werden kann. Daher sollten entsprechende Samples verworfen werden. Abbildung 4.11 zeigt eine demodulierte Spur mit 2 Interrupts, die als große Spitzen erkennbar sind.

Zur Detektion von Interrupts kam ein Schwellwert zum Einsatz, mit dem umliegende Samples einer bestimmten Breite auf NaN gesetzt werden. Dies erlaubt die nachträgliche Anwendung einer STFT, wobei alle Elemente des entsprechenden Zeitabschnittes ebenfalls den Wert NaN aufweisen. Diese Werte können später bei der Mittelwertbildung ignoriert werden und führen somit zu keiner Verfälschung des Ergebnisses. Ein Ersetzen mit anderen Werten würde scharfe Kanten im Signal erzeugen, welche zu breitbandigen Störungen im Spektrogramm führen.

5 OpenSSL

Da RSA auf der Berechnung von sehr großen Zahlen basiert, können diese nicht direkt von der CPU verarbeitet werden. Grund hierfür ist die geringe Wortbreite von modernen CPUs (aktuell idR. 64 bit) im Vergleich zu der Länge des Moduls (idR. 1024 bis 4096 Bit). Eine übliche Darstellung ist die Speicherung als Array von Integer. Die einzelnen Elemente des Arrays, auch Limbs genannt, werden dabei separat behandelt. In OpenSSL ist dies in der Big Number Library (BN) implementiert [SSL]. Alle grundlegenden Operationen wie Addition oder Multiplikation wurden auf diese Datenstruktur angepasst. Diese Implementierung weist dabei Seitenkanaleffekte auf, die im Folgenden untersucht werden. Dabei wird ein Angriff auf die modulare Potenzierungsmethode vorgestellt, welche in der RSA-Implementierung Verwendung findet. Diese Arbeit bezieht sich auf OpenSSL Version 1.0.0.s .

5.1 Seitenkanaleffekte von Multiplikationen

Shamir et al. haben gezeigt, dass in GnuPG Multiplikationen mit bestimmten Zahlen charakteristische Änderungen im Spektrogramm erzeugen. Ist eines der Argumente eine Zahl, die viele Limbs mit dem Wert Null enthält (wie z.B. 2^n), führt dies zu einem unterschiedlichen Frequenzspektrum, als wenn beide Zahlen ein zufälliges Bitmuster aufweisen [GST14]. Dieses Verhalten war wesentlicher Bestandteil der Angriffe [GST14], [GPT14] und [Gen+15]. Um auf dieses Verhalten bei OpenSSL zu prüfen, wird Testprogramm 5.1 benutzt. Dabei ist `r` eine zufällige Zahl und `arg` kann von dem Angreifer frei gewählt werden. Da eine einzelne Multiplikation sehr kurz ist, wird die Multiplikation mehrfach aufgerufen, um die Seitenkanaleffekte zu verstärken. Dieses Verhalten ist ebenfalls bei der binären Potenzierung der Fall, da die Multiplikation in Zeile 8 mehrfach durchlaufen wird. Die Iterationszahlen in Listing 5.1 werden so gewählt, dass alle drei Schleifen etwa dieselbe Laufzeit aufweisen. Sowohl in der rohen als auch in der demodulierten Spur in Abbildung 4.9 sind die Dummyoperationen deutlich von den OpenSSL Multiplikationen unterscheidbar, welche sich zwischen 25ms und 45ms befinden.

```
1 //Dummyoperationen
2 for ( i=0; i < 8000000; i++) i ^= 0;
3
4 //BN_mod_mul(r, r, arg, m) : r = (r * arg) mod N
5 for (i = 0; i < 400; i++) BN_mod_mul(r, r, arg, N);
6
7 //Dummyoperationen
8 for ( i=0; i < 8000000; i++) i ^= 0;
```

Listing 5.1: C Pseudocode für das Testprogramm cprog/openssl-mul.c

Dieses Programm wird einer DPA Analyse unterzogen. Die Spuren werden dabei, wie in Kapitel 4 beschrieben vorverarbeitet. Hierzu wurden die Argumente 2^{4095} und eine zufälligen 4096-Bit-Zahl gewählt, wobei der Modul N 4096-Bit beträgt. Abbildung 5.1 zeigt die Unterschiede im Spektrogramm. Zwischen 20 ms und 35 ms sind klare Unterschiede zwischen den beiden Testgruppen zu erkennen, was auf einen Zusammenhang mit den OpenSSL Multiplikationen

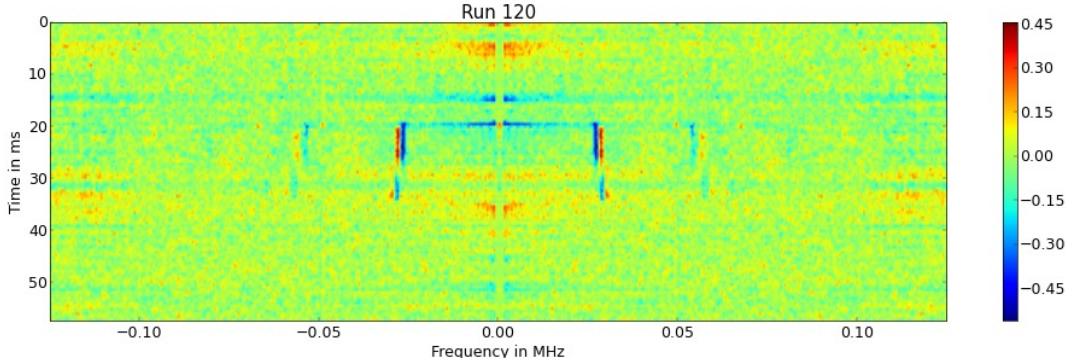


Abbildung 5.1: DPA Spektrogramm von 400 aufeinander folgenden OpenSSL Multiplikationen mit 2^{4095} bzw. einer zufälligen Zahl als Argument. Es wurden 120 Messungen durchgeführt, wobei die Länge des Moduls 4096-Bit beträgt.

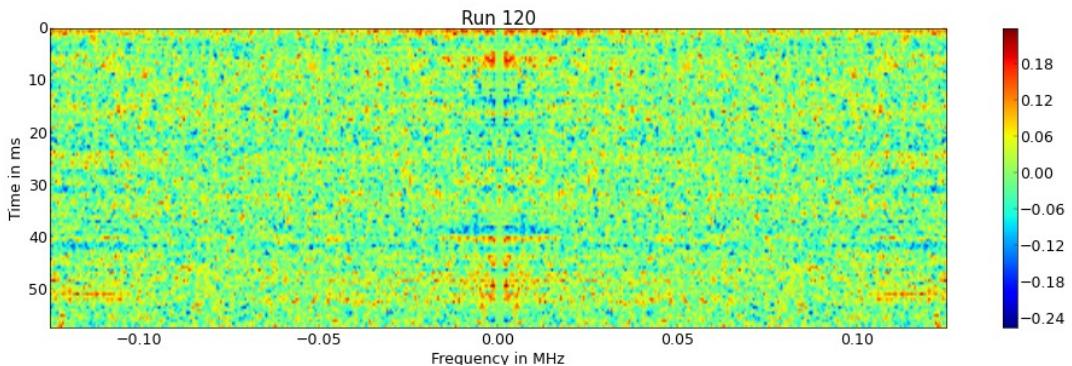


Abbildung 5.2: DPA Spektrogramm von 400 aufeinander folgenden OpenSSL Multiplikationen mit zwei zufälligen Zahlen als Argument. Es wurden 120 Messungen durchgeführt, wobei die Länge des Moduls 4096-Bit beträgt.

schließen lässt. Diese sind in ihrem Frequenzband begrenzt und über die Zeit konstant. Wiederholt man denselben Versuch mit zwei verschiedenen Zufallszahlen (Abbildung 5.1), treten diese Unterschiede nicht auf. Somit lässt sich anhand des DPA-Spektrogramms bestimmen, ob `arg` viele Null Limbs enthält oder nicht.

5.2 Laufzeitverhalten und Seitenkanaleffekte

Eine Erkenntnis ist der Zusammenhang zwischen der Stärke von Seitenkanaleffekten und der Schlüssellänge. Bei vielen Angriffen auf RSA, wie z.B. der Faktorisierung von N , ist eine lange Schlüssellänge problematisch, da der Suchraum zu groß wird. Operationen wie Multiplikationen werden jedoch mit kürzerer Schlüssellänge schneller, wodurch Seitenkanaleffekte schlechter erkennbar werden.

Um diesen Effekt zu zeigen, wird der Versuch aus 5.1 mit einem 2048-Bit an Stelle eines 4096-Bit-Modul wiederholt. Vergleicht man die DPA Spektrogramme 5.1 und 5.3 fällt auf, dass die Seitenkanaleffekte wesentlich kürzer ausfallen. Dies ist im wesentlichen auf die kürzere Laufzeit der Operationen zurückzuführen. Zudem fällt auf, dass sich ebenfalls die Lokalisation des Seitenkanaleffektes auf der Frequenzachse geändert hat. Die Seitenkanaleffekte treten nun

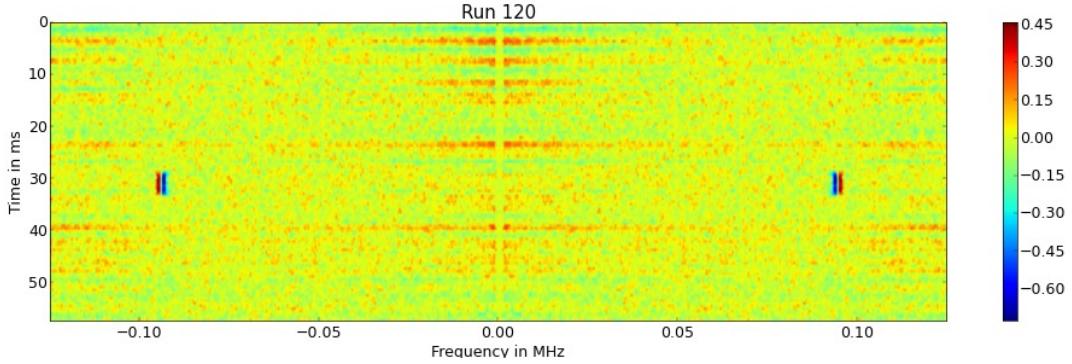


Abbildung 5.3: DPA Spektrogramm von 400 aufeinander folgenden OpenSSL Multiplikationen mit 2^{2047} bzw. einer zufälligen Zahl als Argument. Die Länge des Moduls beträgt 2048-Bit.

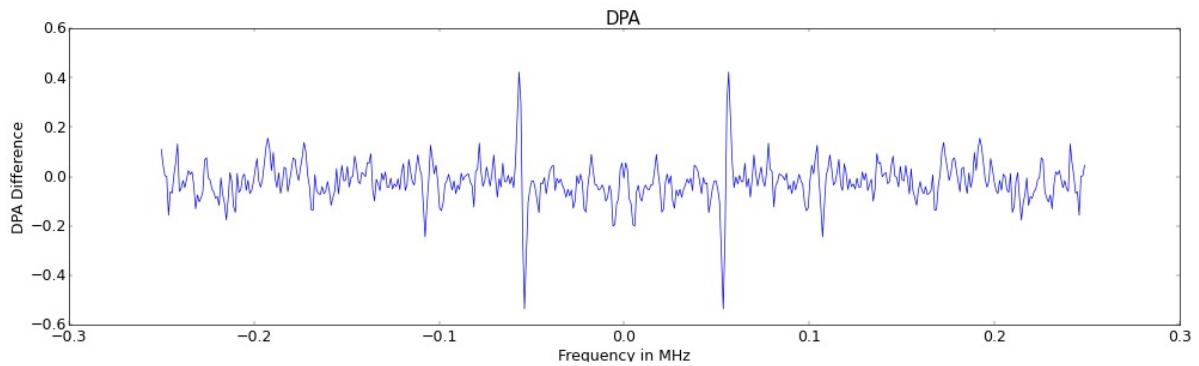


Abbildung 5.4: DPA Differenz zu einem festen Zeitpunkt im DPA Spektrogramm mit einem 4096-Bit-Modul. Dieses wurde aus dem DPA Spektrogramm 5.1 abgeleitet.

bei etwa der dreifachen Frequenz auf, im Vergleich zu einem 4096-Bit-Modul. Dadurch ist es erforderlich, den Angriff für eine bestimmte Schlüssellänge zu kalibrieren.

Abbildung 5.4 zeigt die DPA Differenz zu einem festen Zeitpunkt im Spektrogramm 5.1. Hierbei ist die x-Achse die Frequenz und die y-Achse die Differenz der Intensität. Es ist zu erkennen, dass eine starke positive Abweichung direkt neben einer starken negativen Abweichung auftritt. Dies deutet darauf hin, dass sich ein Frequenzanteil in der demodulierten (Abbildung 4.10 Spur mit der Wahl des Arguments verschiebt. Es wurden Laufzeitmessungen für 1.000.000 Multiplikationen mit einem 4096-Bit-Modul durchgeführt. Für die Argumenten 2^{4095} betrug die Laufzeit 21.5s (46.510 Multiplikationen pro Sekunde) und für eine zufällige 4096-Bit-Zahl 22.5s (44.440 Multiplikationen pro Sekunde). Dabei fällt auf, dass Multiplikationen mit 2^{4095} geringfügig schneller sind, als die mit einer zufälligen Zahl. Dies und die Tatsache, dass sich die Lokalisation des Seitenkanaleffekts mit der Modullänge ändert, lässt vermuten, dass die Seitenkanaleffekte durch Laufzeitdifferenzen bzw. Änderung der Wiederholungsrate der Multiplikationen erzeugt werden. Ein direkter Zusammenhang zwischen der genauen Lokalisation des Seitenkanals auf der Frequenzachse und der Wiederholungsrate konnte dabei nicht gezeigt werden. Dieses Laufzeitverhalten wurde jedoch von [BT11] in ähnlicherweise ausgenutzt, wie in Kapitel 5.4 beschrieben ist.

5.3 OpenSSL - BN_mod_exp

Die Funktion `BN_mod_exp` ist die modulare Potenzierungsmethode von OpenSSL. Im Fall von RSA wird der Aufruf `BN_mod_exp` und die Funktion `BN_mod_exp_mont` deligiert. Eine möglicher Angriff auf den RSA-CRT Modul ist eine binäre Suche und ist in 5.4 beschrieben. Hierzu ist es notwendig zu bestimmen, ob die modulare Reduktion zu Beginn der Berechnung Zeile 4 Listing 5.2 durchgeführt wird. Im Laufe des Angriffs nähert sich die Eingabe c dem Modul N an. Dabei erzeugt sowohl die Multiplikation mit $c \bmod N$ als auch die modulare Reduktion Seitenkanaleffekte. Listing 5.2 zeigt nochmals den Pseudocode der m-Array Potenzierung, welche ähnlich der von OpenSSL verwendeten Sliding Window Variante ist.

```

1 //Berechnung von c^d mod N
2 function m_array_exp(c, d, N)
3     if c > N
4         c = c mod N
5
6     //pow[i] = c^i mod N
7     pow[0] = 1
8     for i = 1...m
9         pow[i] = pow[i-1] * c mod N
10
11    D = Zerlegung von d in m-Bit Wörter
12    k = length(D)
13
14    res = D[k-1]
15    for i = k-2...0
16        res = res ^ (2^m) mod N // Schritt a
17        if D[i] > 0
18            res = res * pow[D[i]] mod N // Schritt b
19
20    return res

```

Listing 5.2: m-Array Methode zur Potenzierung.

Wie in 5.1 beschrieben, lässt sich über den Seitenkanal bestimmen, ob `pow[D[i]]` in der Multiplikation Zeile 18 Listing 5.2 viele Null Limbs enthält. Das Array `pow` enthält dabei die Potenzen $\text{pow}[i] = c^i \bmod N$. Da c in der Größenordnung des Moduls N ist, wird für alle Potenzen $i > 1$ eine Reduktion durchgeführt. Diese weisen dadurch ein zufälliges Bitmuster auf. Somit kann nur das Element $\text{pow}[1] = c \bmod N$ zu diesem Seitenkanaleffekt beitragen. Wählt man für eine DPA Analyse als Argument eine Zahl mit vielen Null Limbs und als Referenz eine zufällige Zahl, können zwei Fälle eintreten:

1. Es treten Seitenkanaleffekte auf und c enthält viele Null Limbs. In diesem Fall wurde c durch die Reduktion in Zeile 4 Listing 5.2 nicht verändert und unverändert in Zeile 8 verwendet.
2. Es treten keine Seitenkanaleffekte auf. Nach der Reduktion in Zeile 4 Listing 5.2 ist $c = c - N$ eine zufällige Zahl, welche kaum Null Limbs enthält.

Entscheidend für die Stärke dieses Seitenkanaleffektes ist die Anzahl der Multiplikationen mit $c \bmod N$. Je mehr Multiplikationen mit diesem Wert durchgeführt werden, desto stärker ist der Seitenkanaleffekt. Im Fall von RSA verwendet OpenSSL eine Fensterbreite $m = 5$. Aus `crypto/bn/bn_lcl.h` Zeile 147 und `crypto/bn/bn_exp.c` Zeile 436 ist ersichtlich, dass ab einem 671-Bit-Modul (1342-Bit RSA) eine Fensterbreite von $m = 5$ Bit verwendet wird. Für die

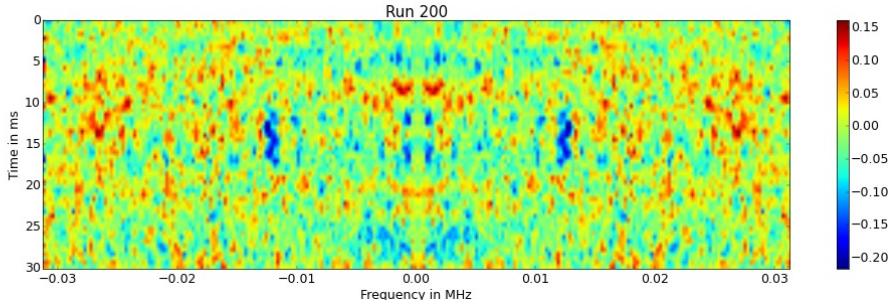


Abbildung 5.5: Das DPA Spektrogramm mit der Kombination der beiden Seitenkanaleffekte, wie sie im Folgenden für den Angriff verwendet werden.

m-Array Methode und einen 2048-Bit-Modul bedeutet dies, dass maximal $\lceil 2048/5 \rceil = 410$ Zugriffe auf das Array `pow` erfolgen können. Die Zugriffsreihenfolge ist dabei abhängig von dem Exponenten und kann als zufällig angesehen werden. Damit werden im Schnitt nur $410/32 \approx 12.8$ Multiplikationen mit $c \bmod N$ durchgeführt. Die Sliding Window Methode, welche bis zu 8% der Multiplikationen einspaart, weist somit ebenfalls etwa 12 Multiplikationen mit diesem Wert auf. Dies führt zu einem wesentlich schwächeren Seitenkanaleffekt als bei 5.1, der erst nach etwa 400 Messungen sichtbar wird.

Einen weiteren Seitenkanaleffekt erzeugt die modulare Reduktion. Diese wird in OpenSSL genau dann ausgeführt, wenn $c < 0$ oder $c > N$ gilt. Ansonsten wird dieser Schritt übersprungen (`bn/crypto/bn_exp.c` Zeile 422-433). Dies führt zu einem zusätzlichem Seitenkanaleffekt, der sich mit dem aus 5.1 addiert. Dieser konnte unabhängig vom Ersten Seitenkanaleffekt nachgewiesen werden und tritt ebenfalls nach etwa 400 Messungen auf. Abbildung 5.5 zeigt die Kombination der beiden Seitenkanaleffekte. Diese sind unregelmäßiger und schwächer als bei 5.1 aber dennoch klar erkennbar. Diese wurden im folgenden Angriff zur Erkennung der Reduktion verwendet und treten nach etwas mehr als 200 Messungen auf.

5.4 Angriff auf OpenSSL

Im Folgenden wird beschrieben, wie der bekannte Angriff auf GnuPG aus [GST14] auf OpenSSL übertragen wird. Der in dieser Arbeit vorgestellte Angriff benutzt die Reduktion $c = c \bmod N$ in Zeile 4 Listing 5.2. Diese kann dazu benutzt werden, um zu die Fälle $c \leq N$ und $c > N$ zu unterscheiden, um eine binäre Suche auf dem Modul durchzuführen. Da bei der Verwendung des privaten Schlüssels die Optimierung durch den Chinesischen Restsatz benutzt wird, beträgt der Modul in diesem Fall p bzw. q . Dies führt zu der Faktorisierung des RSA Moduls N . Bei diesem Angriff werden einzeln die Bits des Moduls vom MSB zum LSB extrahiert. Dabei ist es ausreichend, lediglich die obere Hälfte des Moduls zu bestimmen um den vollständigen Schlüssel zu berechnen [Cop97].

```

1  || function attack()
2      ref = Gro ß e n Bit Zufallszahl
3
4      //Reduktion
5      r = Gro ß e n Bit Zufallszahl
6      red = DPA( r, ref)
7
8      //Keine Reduktion
9      nored = DPA( 2^(n-1), ref)
10
11     //Angriffsphase
12     secret = 2^(n-1)
13     for i = (n-2)...
14         d = DPA(secret + 2^i, ref)
15
16         if |d - red| < |d - nored|
17             secret_i = 0 //secret + (2^i) > p
18         else
19             secret_i = 1 //secret + (2^i) < p
20
21     return secret

```

Listing 5.3: Pseudocode des Angriffs auf den CRT Modul.

Listing 5.3 zeigt den Pseudocode für diesen Angriff. Dieser wird in zwei Schritten durchgeführt, einer Kalibrierungsphase und einer Angriffsphase. Erst findet eine Kalibrierungsphase statt, welche nötig ist, da je nach Signal Rausch Verhältnis und Varianz der Messungen die Mittelwerte unterschiedlich stark konvergieren. Des Weiteren wird angenommen, dass das MSB von N gesetzt ist, so dass $2^n > N > 2^{n-1}$ gilt, wobei n die Länge des Moduls ist. Eine große zufällige Zahl ref wird als Referenzfall gewählt, welche mit sehr hoher Wahrscheinlichkeit zu einer Reduktion führt. Diese wird bei allen folgenden DPA Analysen des Angriffs benutzt. Um wiederholte Messungen mit dem Referenzfall zu vermeiden ist es möglich, den selben Mittelwert für alle folgenden DPA Analysen zu verwenden.

Die Funktion `DPA(a, b)` führt eine DPA Analyse mit den Argumenten `a` und `b` durch. Als Rückgabewert liefert diese die Differenz zwischen Minima und Maxima des DPA Spektrogramms. Dabei wurden nicht relevante Teile des Spektrogramms maskiert, um die Fehleranfälligkeit zu verringern. Diese Differenz dient als Indikator für die Stärke des Seitenkanaleffektes. In der Kalibrierungsphase werden zwei DPA Analysen durchgeführt. Die Anzahl der benötigten Messungen muss dabei so gewählt werden, dass diese Fälle anhand des Indikators klar zu unterscheiden sind.

1. Die Reduktion wird durchgeführt. In dieser DPA Analyse wird der Unterschied von Berechnungen betrachtet, welche stets eine Reduktion durchführen. Hierzu wird eine weite zufällige Zahl r ähnlich dem Referenzfall gewählt, welche stets zu einer Reduktion führt. Obwohl hier keine Unterschiede im Spektrogramm zu erwarten sind, konvergiert dieser Fall je nach Varianz des Eingangssignals unterschiedlich schnell gegen Null.
2. Die Reduktion wird nicht durchgeführt. Dieser Fall zeigt den Unterschied zwischen Berechnungen, bei denen die Reduktion durchgeführt bzw. übersprungen wird. Neben dem Referenzfall, wird als Argument 2^{n-1} gewählt. Da das MSB von N stets gesetzt ist, führt dieser Fall zu keiner Reduktion.

Anschließend findet die Angriffsphase statt, um die einzelnen Bits des Moduls zu extrahieren (Zeile 12-19 Listing 5.3). Die Variable `secret` beinhaltet die bekannten oberen $n \dots i+1$ Bit des Moduls, wobei die restlichen Bit den Wert 0 aufweisen. Zu Beginn des Angriffs wird diese mit dem Wert 2^{n-1} initialisiert, so dass $secret < N$ gilt, jedoch dass MSB gesetzt ist. Diese Bedingung gilt zudem am Anfang jeder Iteration der `for`-Schleife. In Zeile 14 Listing 5.3 wird eine DPA Analyse mit dem Wert $secret + 2^i$ durchgeführt. Anhand des Indikators `d` wird in Zeile 16 Listing 5.3 bestimmt, ob dieser Wert zu einer Reduktion führt.

1. Die Reduktion wird durchgeführt, demnach gilt $secret + 2^i > N$. Da die Bits $i - 1 \dots 0$ den Wert 0 aufweisen, muss das Bit i den Wert 0 aufweisen (Zeile 17 Listing 5.3).
2. Die Reduktion wird nicht durchgeführt, demnach gilt $secret + 2^i < N$. Wäre dieses Bit 0, würden in den folgenden Iterationsschritten der Wertebereich $secret + 2^i \dots secret + 2^{i-1}$ betrachtet werden. Da jedoch $secret + 2^i < N$ gilt, kann sich in diesem Wertebereich der Modul N nicht befinden. Somit ist dieses Bit 1 (Zeile 19 5.3).

Dieser Vorgang wird so lange wiederholt, bis der gesamte Modul bekannt ist.

Im Laufe des Angriffs nähert sich c dem Modul N an. Im Fall einer Reduktion konvergiert $c \mod N$ somit gegen Null und die Länge der Zahl nimmt ab. Die Länge des Referenzfalls $r \mod N$ bleibt jedoch konstant in der Größenordnung von N . Dies erzeugt zeitliche Verschiebungen in den Berechnungen und somit zu unerwünschten Seitenkanaleffekten. Da im Fall der Reduktion keine Seitenkanaleffekte auftreten dürfen, führt dies u.U. zu einer Fehlerkennung eines Bits. Um dieses Problem zu umgehen, kann im Verlauf des Angriffs als Referenzfall der Wert $secret + 2^{i+1}$ gewählt werden. Diese Zahl führt in jedem Fall zu einer Reduktion und befindet sich in der selben Größenordnung wie $secret + 2^i$ nach einer Reduktion.

6 Fazit

Im Allgemeinen konnte gezeigt werden, dass OpenSSL ähnliche Seitenkanaleffekte wie GnuPG aufweist. Ein direkter Vergleich der beiden Implementierungen wurde jedoch nicht vorgenommen. Dennoch konnten die Erkenntnisse aus [GST14] und [Gen+15] teilweise auf OpenSSL übertragen werden. Für die hier vorgestellten Versuche wurde ein Dell Optiplex 7010 SFF verwendet [Inc].

Für die durchgeführten Versuche war es erforderlich, das Gehäuse des DUT zu öffnen, um den Seitenkanal zu messen. Die elektromagnetische Strahlung würde sonst durch das Metallgehäuse weitgehend abgeschirmt werden. Dies limitiert den praktischen Nutzen dieses Angriffs mit einer Antenne für Server und Desktop PCs, welche nicht ein Metallgehäuse besitzen. Dennoch ist ein Messen des Seitenkanals über den USB-Port mitunter möglich. Hierfür ist jedoch physischer Kontakt mit DUT notwendig. Viele Laptops hingegen besitzen ein Kunststoffgehäuse, welches kaum einen Schutz vor diesem Angriff bietet [Gen+15].

Um Störungen durch andere Programme zu vermeiden, wurden alle Tests mit einer minimalen Ubuntu Installation durchgeführt. Diese Annahme ist in der Praxis jedoch nicht korrekt, da auf dem Zielsystem unter Umständen weitere Programme laufen. Je nach Anwendung kann es dabei zu unterschiedlich starken Störungen kommen. Empirische Versuche haben gezeigt, dass ein Webbrowser bereits genug Störungen erzeugt, sodass bereits eine Ausrichtung der Spuren mit dem hier vorgestelltem Verfahren nicht möglich ist. Abbildung 6.1 zeigt die zur Ausrichtung verwendete Intensität einer Trägerfrequenz, welche durch Google Chrome gestört wird (siehe 4.5). Dies ist im Wesentlichen durch die große Zahl an Interrupts zu erklären.

Die hier vorgestellten Versuche wurden ebenfalls auf einem Desktop-PC Intel I7 950 und einem Gigabyte X58A-UD3R Mainboard. Dabei wurden abweichende Ergebnisse im Vergleich zu dem Dell Optiplex 7010 SFF festgestellt. In diesem Fall traten Seitenkanaleffekte bereits ohne Frequenzdemodulation auf. Abbildung 6.2 Zeigt das Resultat des Versuchs analog zu 5.1. Diese sind als Seitenbänder einiger Trägerfrequenzen erkennbar. Dies deutet darauf hin, dass in diesem Fall der Seitenkanaleffekt amplitudenmoduliert übertragen wird.

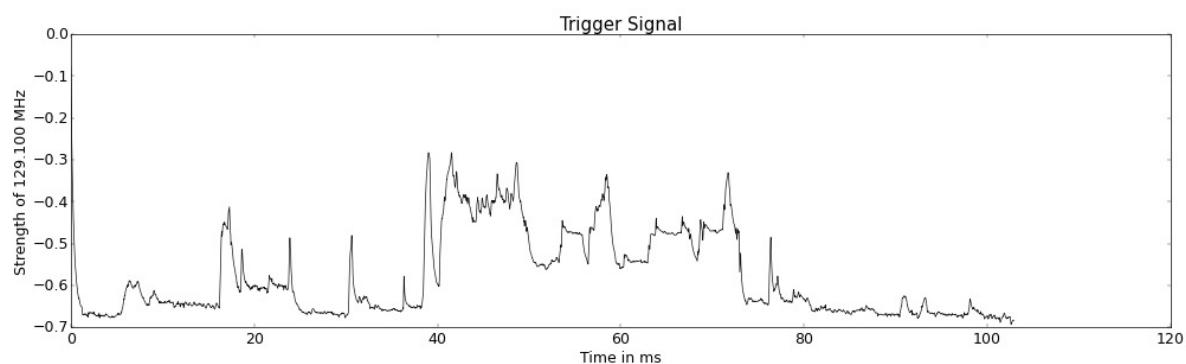


Abbildung 6.1: Die Intensität einer Trägerfrequenz, welche zur Ausrichtung verwendet wurde. Diese Spur wurde durch Google Chrome stark gestört.

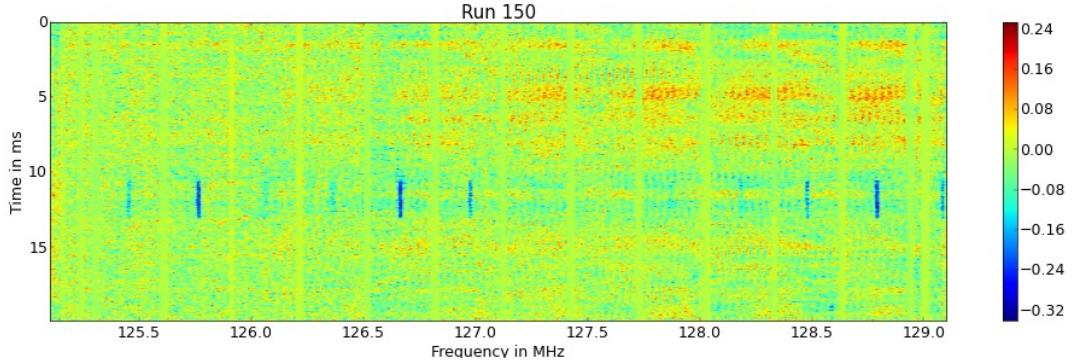


Abbildung 6.2: Seitenkanaleffekt auf einem anderen DUT treten bereits ohne Frequenzdemodulation auf. Die Bandbreite dieses Signal beträgt 4 MHz.

Der Angriff wurde auf die Methode `bn_mod_exp` durchgeführt, welche im Testprogramm an die Funktion `bn_mod_exp_mont` delegiert. Es wurde erfolgreich 176-Bit des Moduls extrahiert, bevor eine Fehlerkennung zum Abbruch des Angriffs geführt hat. Der Modul bestand dabei aus einer 2048-Bit Sequenz der Zahl `0xdeadbeef`. Dabei wurde im Schnitt ein Bit in ca 4 Minuten extrahiert. Der größte Overhead ist dabei die Kommunikation mit dem SDR und die Ausrichtung der Spuren. Ein Video zu diesem Angriff befindet sich auf dem beigelegten Datenträger.

Der vorgestellte Angriff extrahiert die Bits des CRT Modul iterativ vom MSB zum LSB. Es wird für jedes Bit ein Seitenkanalangriff durchgeführt, um dessen Wert zu bestimmen. Da der bekannte Teil des Moduls für den nächsten Iterationsschritt benötigt wird, führt eine fehlerhafte Erkennung eines Bits in den weiteren Iterationsschritten zu Folgefehlern. Wird z.B. ein Bit fälschlicherweise als 1 detektiert gilt in allen folgenden Schritten $c > p$, sodass die Reduktion immer ausgeführt wird. Da mehrere hundert Bits in Folge richtig erkannt werden müssen, erfordert dies eine sehr geringe Fehlerrate.

Der von [Gen+15] vorgestellte Angriff hatte zum Ziel, die Zugriffsreihenfolge auf das Array `pow` in Listing 2.2 zu bestimmen. Hierzu wurden mehrere Geheimtexte so gewählt, dass jeweils die ersten Elemente von `pow` die in 5.1 vorgestellte Eigenschaft besitzen. Dieser Angriff erfordert jedoch eine sehr genaue Ausrichtung sowie eine aufwendige Nachbearbeitung der Spuren. Anschließend waren die Zugriffe auf die maskierten Elemente in der rohen Spur sichtbar. Diese Ergebnisse konnten jedoch nicht reproduziert werden.

Standardgemäß verwendet OpenSSL Blinding, sodass der Angreifer keine Kontrolle über die Eingabe der RSA Entschlüsselung besitzt. Da dies jedoch eine Grundvoraussetzung für den Angriff ist, ist dieser Angriff in der Praxis nicht möglich. Die gezeigten Seitenkanaleffekte beziehen sich dabei auf die Big Number-Library von OpenSSL. Dadurch ist nicht auszuschließen, dass die Implementierung des Blindings selbst Seitenkanaleffekte erzeugt, die ausgenutzt werden können. Dies wurde in dieser Arbeit jedoch nicht untersucht.

Literatur

- [BT11] Billy Bob Brumley und Nicola Tuveri. „Remote timing attacks are still practical“. In: *Computer Security–ESORICS 2011*. Springer, 2011, S. 355–371.
- [Cop97] Don Coppersmith. „Small solutions to polynomial equations, and low exponent RSA vulnerabilities“. In: *Journal of Cryptology* 10.4 (1997), S. 233–260.
- [Cor09a] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009, S. 956–958.
- [Cor09b] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009, S. 899–905.
- [For08] Behrouz A. Forouzan. *Cryptography & Network Security*. 1. Aufl. New York, NY, USA: McGraw-Hill, Inc., 2008, S. 285–298. ISBN: 0073327530, 9780073327532.
- [Gen+15] Daniel Genkin u. a. *Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation*. Cryptology ePrint Archive, Report 2015/170. <http://eprint.iacr.org/2015/170.pdf>. 2015.
- [GPT14] Daniel Genkin, Itamar Pipman und Eran Tromer. „Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs“. English. In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Hrsg. von Lejla Batina und Matthew Robshaw. Bd. 8731. Lecture Notes in Computer Science. Ausführliche Version <http://www.cs.tau.ac.il/~tromer/papers/handsoff-20140731.pdf>, Abgerufen am 14.9.2015. Springer Berlin Heidelberg, 2014, S. 242–260. ISBN: 9783662447086. DOI: 10.1007/978-3-662-44709-3_14. URL: http://dx.doi.org/10.1007/978-3-662-44709-3_14.
- [GST14] Daniel Genkin, Adi Shamir und Eran Tromer. „RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis“. English. In: *Advances in Cryptology – CRYPTO 2014*. Hrsg. von JuanA. Garay und Rosario Gennaro. Bd. 8616. Lecture Notes in Computer Science. Ausführliche Version <http://www.cs.tau.ac.il/~tromer/papers/acoustic-20131218.pdf>, Abgerufen am 14.9.2015. Springer Berlin Heidelberg, 2014, S. 444–461. ISBN: 9783662443705. DOI: 10.1007/978-3-662-44371-2_25. URL: http://dx.doi.org/10.1007/978-3-662-44371-2_25.
- [Inc] Dell Inc. „Dell Optiplex 7010 Technical Guidebook“. In: (). http://www.dell.com/downloads/global/products/optix/en/optiplex_7010_technical_guidebook.pdf Abgerufen am 14.9.2015.
- [KJJ99] Paul Kocher, Joshua Jaffe und Benjamin Jun. „Differential Power Analysis“. English. In: *Advances in Cryptology — CRYPTO' 99*. Hrsg. von Michael Wiener. Bd. 1666. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, S. 388–397. ISBN: 9783540663478. DOI: 10.1007/3-540-48405-1_25. URL: http://dx.doi.org/10.1007/3-540-48405-1_25.
- [Kle+10] Thorsten Kleinjung u. a. „Factorization of a 768-bit RSA modulus“. In: *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, S. 333–350.
- [Knu68] Donald E. Knuth. *The Art of Computer Programming 2 - Seminumerical Algorithms*. 3. Aufl. 1968, S. 285–298. ISBN: 0201896842.

- [Koç95] Cetin K Koç. „Analysis of sliding window techniques for exponentiation“. In: *Computers & Mathematics with Applications* 30.10 (1995), S. 17–24.
- [Noo] NooElec. *Ham It Up - RF Upconverter For Software Defined Radio*. Abgerufen am 14.9.2015. <http://www.nooelec.com/store/ham-it-up.html>.
- [OS06] Yossi Oren und Adi Shamir. *How Not to Protect PCs from Power Analysis*. Rump Session, Crypto 2006. <http://iss.oy.ne.ro/HowNotToProtectPCsFromPowerAnalysis> Abgerufen am 14.9.2015. 2006.
- [Res10] Ettus Research. *USRP B200/B210 Product Overview*. Abgerufen am 14.9.2015. http://www.ettus.com/content/files/kb/b200-b210_spec_sheet.pdf, 2010.
- [RSA78] Ronald L Rivest, Adi Shamir und Len Adleman. „A method for obtaining digital signatures and public-key cryptosystems“. In: *Communications of the ACM* 21.2 (1978), S. 120–126.
- [RTLa] RTL-SDR. *Gnuradio Wiki*. <http://gnuradio.org/redmine/projects/gnuradio/wiki> Abgerufen am 14.9.2015.
- [RTLb] RTL-SDR. *Osmocom RTL-SDR Technische Informationen*. <http://sdr.osmocom.org/trac/wiki/rtl-sdr> Abgerufen am 14.9.2015.
- [RTLc] RTL-SDR. *RTL-SDR Projekt Homepage, About RTL-SDR*. <http://www rtl-sdr.com/about-rtl-sdr/> Abgerufen am 14.9.2015.
- [Sch+10] O Schimmel u. a. „Correlation power analysis in frequency domain“. In: *COSADE 2010 First International Workshop on Constructive SideChannel Analysis and Secure Design*. 2010.
- [Sel05] Ivan Selesnick. „Short time fourier transform“. In: *Connexions*, Aug 9 (2005).
- [SF08] GN Shinde und HS Fadewar. „Faster RSA algorithm for decryption using Chinese remainder theorem“. In: *ICCES: International Conference on Computational & Experimental Engineering and Sciences*. Bd. 5. 4. 2008, S. 255–262.
- [Sig] Bekanntmachung zur elektronischen Signatur. „Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen“. In: (). http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/2013Algorithmenkatalog.pdf?__blob=publicationFile Abgerufen am 14.9.2015, S. 7.
- [Smi99] Douglas C Smith. „Signal and noise measurement techniques using magnetic field probes“. In: *Electromagnetic Compatibility, 1999 IEEE International Symposium on*. Bd. 1. IEEE. 1999, S. 559–563.
- [SS01] Rainer Schlittgen und Bernd HJ Streitberg. *Zeitreihenanalyse*. Oldenbourg Verlag, 2001, S. 44, 45.
- [SSL] Open SSL. *OpenSSL Big Number Library Documentation*. <https://www.openssl.org/docs/manmaster/crypto/bn.html> Abgerufen am 14.9.2015.
- [Sta10] Francois-Xavier Standaert. „Introduction to Side-Channel Attacks“. English. In: *Secure Integrated Circuits and Systems*. Hrsg. von Ingrid M.R. Verbauwheide. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, S. 27–42. ISBN: 9780387718293. DOI: 10.1007/978-0-387-71829-3. URL: <http://dx.doi.org/10.1007/978-0-387-71829-3>.

- [Ste] Ing Eckehard Steinbach. „Digitalisierung als Grundlage des Informationszeitalters“. In: *Muenchner-wissenschaftstage* ().
- [SV93] M. Shand und J. Vuillemin. „Fast implementations of RSA cryptography“. In: *Computer Arithmetic, 1993. Proceedings., 11th Symposium on*. Juni 1993, S. 252–259. doi: 10.1109/ARITH.1993.378085.

7 Anhang

7.1 Inhalt der DVD

Auf der beigelegten DVD befindet sich eine digitale Kopie dieser Arbeit und der verwendete Quelltext. Des Weiteren sind dort mehrere Datensätze von Spuren und Konfigurationsdaten, mit denen die Implementierung getestet werden kann. Hierzu ist der Aufruf `python dpa.py/Messungen/testname.json` aus dem Quelltextverzeichnis erforderlich. Zum Start des Programms ist eine GNURadio Version ab 3.7 nötig, welche sich im Fall von Ubuntu im Repository befindet. Die Ergebnisse der DPA Analysen befinden sich ebenfalls auf dem Datenträger. Da der Angriff auf den Modul eine Interaktion mit dem Testprogramm erfordert, sind hierzu keine Datensätze, jedoch ein Video des Angriffs vorhanden. Die folgenden Testfälle befinden sich auf der DVD:

`mul-2n` Datensatz zu 5.1 mit einem 2048-Bit Modul. Die Argumente sind $2^{2047} + 2^{2046}$ bzw. eine zufällige 2048-Bit Zahl. Hierbei sind Unterschiede in den Spektrogrammen zu erwarten. Das verwendete Tesprogramm ist `cprog/openssl-mul.c`.

`mul-rand` Datensatz zu 5.1 mit einem 2048-Bit Modul. Die Argumente sind zwei zufällige 2048-Bit Zahlen. Hierbei sind keine Unterschiede in den Spektrogrammen zu erwarten. Das verwendete Tesprogramm ist `cprog/openssl-mul.c`.

`mul-4096-2n` Analog zu `mul-2n` mit einem 4096-Bit Modul. Das verwendete Tesprogramm ist `cprog/openssl-mul-4096.c`.

`mul-4096-rand` Analog zu `mul-rand` mit einem 4096-Bit Modul. Das verwendete Tesprogramm ist `cprog/openssl-mul-4096.c`.

`exp-2n` Datensatz zu 5.3 mit einem 2048-Bit Modul, welcher aus Wiederholung der Sequenz `0xdeadbeef` besteht. Die Argumente betragen 2^{2047} und eine zufällige 2048-Bit Zahl, die zu keiner Reduktion führt. Hierbei sind Unterschiede in den Spektrogrammen zu erwarten, die jedoch erst nach sehr vielen Spuren auftreten. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

`exp-rand` Analog zu `exp-2n`, wobei beide Argumente zufällige Zahlen sind, die nicht zu einer Reduktion führen. Hierbei sind keine Unterschiede in den Spektrogrammen zu erwarten. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

`exp-nored` Datensatz zu 5.3 mit einem 2048-Bit Modul, welcher aus Wiederholung der Sequenz `0xdeadbeef` besteht. Die Argumente sind zwei zufällige 2048-Bit Zahlen, die zu einer Reduktion bzw. zu keiner Reduktion führen. Hierbei sind Unterschiede in den Spektrogrammen zu erwarten, die jedoch erst nach sehr vielen Spuren auftreten. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

`exp-red` Analog zu `exp-nored`, wobei beide Zahlen zu einer Reduktion führen. Hierbei sind keine Unterschiede in den Spektrogrammen zu erwarten. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

`attack-red` Datensatz zu 5.4 für ein 2048-Bit Modul. Dieser Fall tritt ein, falls eine Reduktion durchgeführt wird. Hierbei sind keine Unterschiede in den Spektrogrammen zu erwarten. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

`attack-nored` Datensatz analog zu `orcale-red`. Dieser Fall tritt ein, falls die Reduktion nicht durchgeführt wird. Hierbei sind Unterschiede in den Spektrogrammen zu erwarten, die früher erkennbar sind als bei `exp-nored` und `exp-red`. Das verwendete Tesprogramm ist `cprog/openssl-exp.c`.

7.2 SDR Empfänger

Als Hardware kam ein USRP B200 von Ettus Research zum Einsatz. Dieser Empfänger ist in der Lage, in dem Frequenzbereich von 70MHz bis 6GHz zu arbeiten. Die Abtastrate beträgt bis zu 32MHz bei 12-Bit Auflösung [Res10]. Erfolgreiche Seitenkanalangriffe wurden in wesentlich tieferen Frequenzbereichen (<3MHz) durchgeführt [Gen+15] und [GPT14]. Die in dieser Arbeit genutzten Signale befinden sich ebenfalls in diesem Frequenzbereich. Grund hierfür ist, dass hochfrequente Störungen schon durch einfache Bauteile stark abgeschwächt werden können [Gen+15]. Um solche Signale dennoch zu messen, ist ein sog. Upconverter nötig, der das Signal in einen höheren Frequenzbereich verschiebt. Hierzu wird der 'Ham It Up' Upconverter benutzt, welcher das Eingangssignal um 125MHz verschiebt. Dies bedeutet, dass ein Signal von 2MHz anschließend auf 127Mhz empfangbar ist [Noo].

7.3 Implementierung

Im Folgenden wird die Implementierung des Angriffs beschrieben. Einige Module lassen sich dabei als Programm starten, um Parameter zu konfigurieren bzw. eine DPA Analyse durchzuführen. Alle Programme erwarten dabei eine Konfigurationsdatei im JSON Format als Argument. Zu jedem Modul sind ebenfalls die notwendigen Konfigurationsparameter beschrieben. Hilfsfunktionen wie das STFT und das Darstellen von Graphen sind in der Datei `dsp.py` implementiert. Für die Berechnung einer FFT wurde die Implementierung von Numpy verwendet.

7.3.1 DUT

Die Datei `dut.py` implementiert die Netzwerkkommunikation zwischen dem DUT und dem Angreifer. Die Klasse `dut_service` implementiert dabei einen TCP Server, welcher auf Anfragen des Clients mit dem Namen des Testprogramms und Argumenten wartet. In dieser Datei sind ebenfalls die Testprogramme und deren Argumente für eine DPA Analyse definiert. Der Client ist in der Klasse `dut` definiert und implementiert eine Methode `challenge(self, challenge)` um eine Anfrage an das DUT zu senden. Um die vorhandene Implementierung auf einen neuen Testszenario anzupassen, ist lediglich eine neue Implementierung dieser Methode erforderlich. Zur Konfiguration ist das zu verwendende Testprogramm als `dut.cmd` und die IP des DUT `dut.ip` und der port `dut.port` anzugeben. Wird ein Testlauf mit dem SDR durchgeführt, muss dieses Programm auf dem DUT ausgeführt werden.

7.3.2 Capture

Die Kommunikation mit dem SDR und die Extraktion der Berechnung erfolgt in der Klasse `capture` aus dem Modul `capture.py`. In diesem ist ebenfalls das Speichern und Wiedergeben von Spuren umgesetzt. Um Speicherplatz zu sparen und den Schritt der Extraktion nicht wiederholt durchzuführen, werden nur die ausgerichteten Spuren gespeichert. Die Methode `capture()` gibt dabei eine Spur als Zeitserie und das übergebene Argument zurück. Zur Kalibrierung des Triggers lässt sich dieses Datei direkt starten, welche das Triggersignal als Zeitserie und die rohe und ausgerichtete Spur als Spektrogramm darstellt.

Die Datei `capture.grc` enthält einen Gnuradio Flussgraphen, welcher die Kommunikation zwischen Python und der SDR Hardware erlaubt. Dieser Graph wird mithilfe des Gnuradio-Companion in ein Python Modul (`top_block.py`) übersetzt, welches eine API, z.B. zum Setzen von Variablen oder Starten des Graphen, bietet.

Zur Konfiguration der Aufnahme ist es erforderlich, die Mittelfrequenz `capture.frequency` und die Bandbreite `capture.sample_rate` anzugeben. Für die Ausrichtung ist es erforderlich, eine Trägerfrequenz als Triggersignal zu wählen `capture.trigger.frequency`. Da hierzu die Anwendung einer STFT erforderlich ist, sind ebenfalls die Parameter `capture.trigger.fft_len` und `capture.trigger.fft_step` erforderlich. Ein Unterschreiten der Intensität von dem Wert `capture.trigger.threshold`, löst den Trigger aus.

7.3.3 Vorverarbeitung

Demodulation und Erkennung von Interrupts sind in der Datei `preprocess.py` implementiert. Der hierzu verwendete Flowgraph findet sich unter `grc/demod/demod.grc` und kann mit dem Gnuradio-Companion geöffnet und editiert werden (Abbildung 7.1). Diese wird in das Modul `top_block.py` übersetzt, welcher von dem Modul `preprocess.py` eingebunden wird. Dort findet ebenfalls die Konfiguration des Flowgraphen statt. Die Methode `demod_grc(trace)` wird benutzt, um eine Spur als Numpy Array an den Flowgraphen zu übergeben und liefert als Ergebnis die demodulierte Spur.

Die Konfiguration des Flowgraphen findet über Selektoren statt, welche unter `preprocess.demod_select` angegeben werden. Hierbei handelt es sich um ein Tripel mit der folgenden Semantik:

1. Der Haupt-Selektor kann folgende Werte annehmen
 - 0 Die Demodulation wird übersprungen und die Spuren unverändert durchgeleitet.
 - 1 Die Spuren werden um die Frequenz `preprocess.demod_frequency` neu zentriert.
 - 2 Auf die zentrierten Spuren wird ein Tiefpassfilter angewendet mit einer Grenzfrequenz von `preprocess.demod_lowpass`
 - 3 Die Demodulation wird auf die zentrierten und gefilterten Spuren angewendet.
2. Hier wird die Demodulationsart angegeben, wobei 0 für Frequenz- und 1 für Amplitudemodulation steht.

3. Anschließend ist es möglich über den Wert 1 einen Bandpassfilter anzuwenden. Die Parameter für diesen sind `preprocess.demod_bandpass_low` und `preprocess.demod_bandpass_high`.

Anschließend wird die Samplerate um den Faktor `preprocess.demod_decimation` verringert. Werden keine Filter angewendet, ist hier der Wert 1 anzugeben.

Für die Interrupterkennung ist ein Schwellwert `preprocess.interrupt_threshold` erforderlich. Des Weiteren wird ein Parameter für die Breite des Interrupts in Samples `preprocess.interrupt_width` benötigt. Diese beziehen sich auf die Zeitserie vor Anwendung des STFT. Werden in einer Spur mehr als `preprocess.interrupt_max` interrupts detektiert wird die Spur verworfen.

Als letzter Schritt erfolgt die Anwendung einer STFT. Hierfür sind die Parameter `preprocess.fft_len` und `preprocess.fft_step` erforderlich. Mithilfe von `preprocess.stft_log` ist es möglich die nachträgliche Anwendung eines Logarithmus durchzuführen. Dies ist auf Grund des großen Dynamikbereich der STFT sinnvoll.

7.3.4 Implementierung DPA

Die Logik für einen DPA Angriff ist in der Klasse `dpa` in `dpa.py` implementiert. Um eine Spur zu einer Testgruppe hinzuzufügen, wird die Methode `add(group, trace)` benutzt. Diese ist in der Lage, auf Numpy Arrays beliebiger Dimension zu arbeiten, wobei alle Elemente unabhängig betrachtet werden. Da während der Vorverarbeitung von der Interruptbehandlung einzelne Samples verworfen und auf `NaN` gesetzt wurden, ist hier eine Behandlung dieses Falls notwendig. Der Mittelwert ist wie folgt definiert:

$$E(A) = \frac{1}{n} \sum_{i=1}^n a_i$$

Um diesen Fall abzudecken, wurde der Parameter n ebenfalls als Matrix implementiert und hält zu jeder Summe die Anzahl der gültigen Samples. Hierzu bietet Numpy die Möglichkeit, Boolesche Masken zu verwenden, um Operationen wie Addition nur auf bestimmte Elemente einer Matrix anzuwenden. Um eine DPA Analyse durchzuführen, lässt sich dieses Modul ebenfalls direkt mit einer Konfigurationsdatei als Argument starten. Hierbei werden die `dut.py` angegebenen Argumente verwendet.

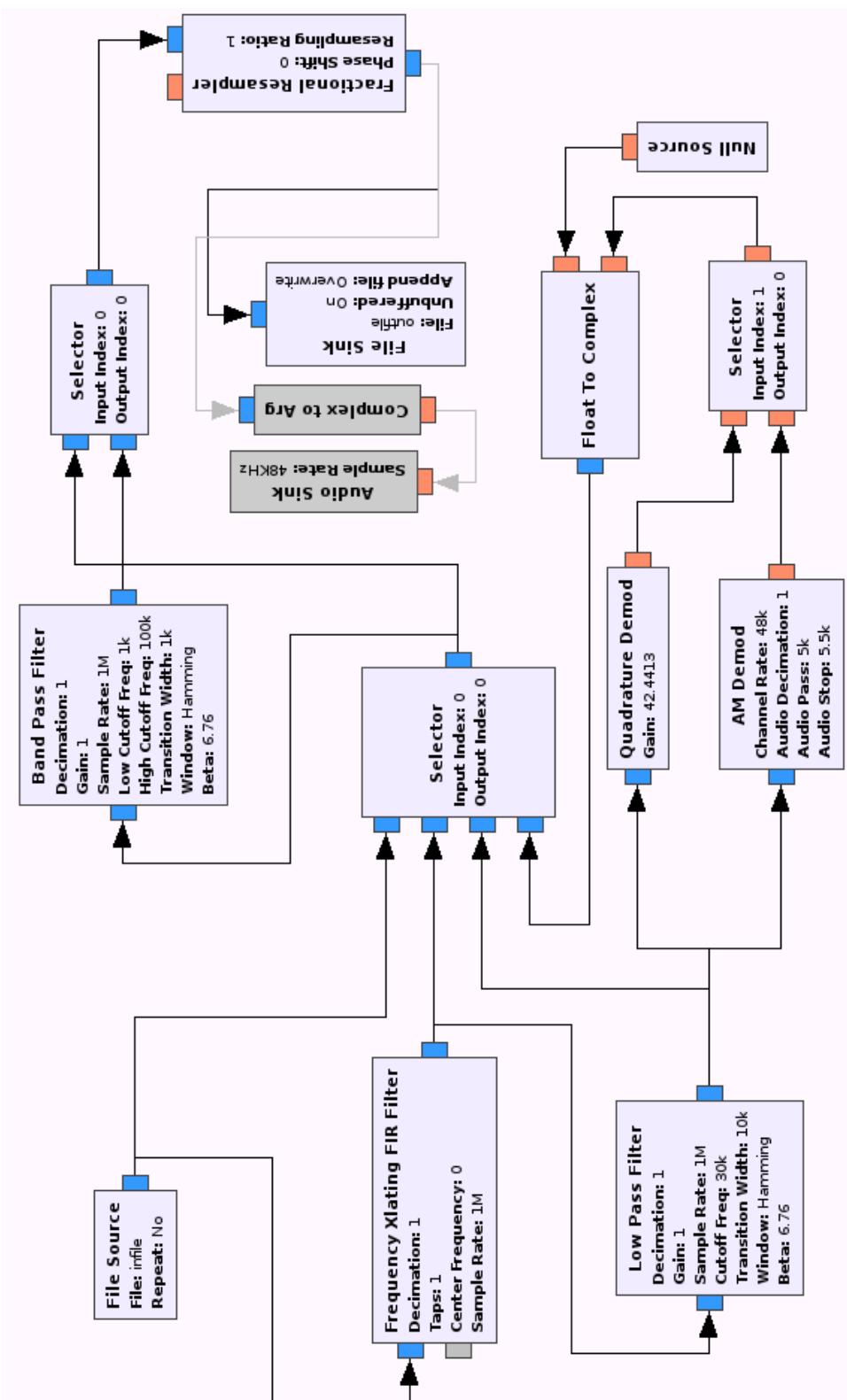


Abbildung 7.1: Dies ist der für die Demodulation verwendete Flowgraph. Das Signal wird von 'File Source' gelesen. Die demodulierte Spur wird mit 'File Sink' geschrieben. Die Nummerierung der Eingänge der Selektoren erfolgt in aufsteigender Reihenfolge von oben nach unten.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg 15.9.2015, Jan Sönke Ruge

Einverständniserklärung

Hiermit erkläre ich mich mit der Aufnahme dieser Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik der Universität Hamburg einverstanden.

Hamburg 15.9.2015, Jan Sönke Ruge