

# Master-Thesis

## Angriffe gegen TPM basierte Festplattenverschlüsselung

Sebastian Lasogga



## Danksagung

Ohne die Unterstützung meines Umfeldes hätte diese Arbeit in der Form nicht angefertigt werden können. Es ist mir deshalb ein innerstes Bedürfnis, mich an dieser Stelle bei allen Personen zu bedanken, die durch ihre Unterstützung zur Entstehung beigetragen haben.

Einen besonderen Dank gilt hierbei Herrn Prof. Dr.-Ing. habil. Ahrens, der mir diese Arbeit überhaupt erst ermöglicht hat. Sein großes Interesse an dem Thema, die geführten fachlichen Gespräche und Anregungen, die zum Gelingen dieser Arbeit geführt haben.

Weiterhin möchte ich dem gesamten Team des LKA Berlin 71 danken. Allen voran Herrn Dipl.-Ing. (FH) Gero Gebert für die Unterstützung während des gesamten Studiums und Herrn Dipl.-Ing. Stefan Hoheisel, der mich mit seinem handwerklichen Geschick beim Kontaktieren des TPM unterstützt hat.

Zuletzt gilt ein besonderes Dankeschön meiner Familie. Ich danke euch für die aufgebrachte Geduld mit mir und für die Zeit, die ihr auf mich verzichten musstet. Ich widme diese Arbeit meinem Sohn Arne. Falls du diese einmal lesen solltest, wünsche ich mir, dass du dabei etwas von der Begeisterung und Neugierde für dein Leben mitnehmen kannst, die ich beim Schreiben dieser Arbeit empfand.

## **Aufgabenstellung**

Im Rahmen dieser Masterarbeit sollen Angriffsvektoren gegen Trusted Platform Modul (TPM) basierte Festplattenverschlüsselungen untersucht und miteinander verglichen werden. Hierzu werden zunächst der Aufbau und die Aufgabe des TPM in einem Computersystem erläutert.

Am Beispiel einer ausgewählten Festplattenverschlüsselung mit TPM soll die Funktionsweise untersucht werden. Dabei werden mögliche Angriffsvektoren identifiziert und vorgestellt. Die Angriffsvektoren sollen hinsichtlich ihrer Erfolgchancen und Rahmenbedingungen miteinander verglichen werden. Anschließend wird der erfolgversprechendste Angriff bestimmt und praktisch validiert.

Zur praktischen Validierung muss ein Versuchsaufbau entwickelt werden. Weitere Randbedingungen, die einen Einfluss auf den Angriff besitzen, sollen anhand dieses Versuchsaufbaus analysiert werden. Der Versuchsaufbau wird die Einrichtung von Zielsystemen, Anpassung der Hardware und Erstellung eigener Software beinhalten.

## **Kurzreferat**

Das Thema TPM basierte Festplattenverschlüsselung hat mit der Einführung von Windows 11 an Bedeutung gewonnen. In dieser Arbeit wurden drei verschiedene Angriffsvektoren gegen die Festplattenverschlüsselung BitLocker vorgestellt. Von den vorgestellten Angriffen zeigt das TPM-Bus-Sniffing die größten Erfolgchancen, weshalb dieser in einer eigenen Versuchsreihe weiter untersucht wurde. Bei diesen Versuchen konnte festgestellt werden, dass der Angriff bei Einhalten bestimmter Randbedingungen (Dekodierung, Abtastrate, erzeugte Datenmenge) zum Erfolg führt. Als Teil dieser Arbeit wurde das Programm ARNE entwickelt, das erfolgreich die aufgezeichnete Kommunikation zwischen dem System und dem TPM dekodieren kann. Es konnte gezeigt werden, dass TPM basierte Festplattenverschlüsselung erfolgreich angegriffen und entschlüsselt werden können.

## **Abstract**

The topic of TPM based hard disk encryption has gained importance with the introduction of Windows 11. In this master thesis, three different attack vectors against the hard disk encryption BitLocker were presented. Of the attacks presented, TPM-bus-sniffing has the greatest chance of success, which is why it was further investigated in a separate series of tests. In these tests, it was found that the attack is successful if certain boundary conditions are met (decoding, sampling rate, amount of data generated). As a part of this work, the program ARNE was developed, which can successfully decode the recorded communication between the system and the TPM. It could be shown that TPM based hard disk encryption can be successfully attacked and decrypted.

# Inhaltsverzeichnis

1	Einleitung und Motivation .....	1
2	Theoretische Grundlagen.....	3
2.1	Trusted Platform Modul (TPM).....	3
2.1.1	Aufbau .....	4
2.1.2	Plattformintegrität und Integritätsmessung .....	6
2.1.3	Schlüsselverwaltung.....	10
2.1.4	Schutzmaßnahmen.....	13
2.1.4.1	Reset Attack Mitigation .....	13
2.1.4.2	Parameter Encryption .....	14
2.1.4.3	Anti-Hammering .....	14
2.1.5	Ausführungsvarianten.....	15
2.1.6	Anwendungen.....	16
2.2	Festplattenverschlüsselungen.....	16
2.2.1	Vergleich Softwarebasierter Festplattenverschlüsselung .....	17
2.2.2	Der BitLocker Algorithmus .....	20
2.3	Theoretische Angriffsvektoren .....	28
2.3.1	Brute-Force gegen den Wiederherstellungsschlüssel.....	29
2.3.2	FVEK im Arbeitsspeicher .....	30
2.3.2.1	DMA Angriff.....	30
2.3.2.2	Cold Boot Angriff .....	31
2.3.3	Mitlesen am TPM.....	32
2.3.3.1	Low Pin Count (LPC) .....	33

2.3.3.2	Enhances Serial Peripheral Interface (eSPI).....	41
2.4	Bewertung der Schwachstellen.....	50
2.4.1	Brute-Force gegen Wiederherstellungsschlüssel.....	50
2.4.2	Cold Boot Angriff.....	51
2.4.3	DMA-Angriff.....	53
2.4.4	TPM Bus-Sniffing.....	53
2.4.5	Zusammenfassung.....	54
3	Praktische Validierung.....	56
3.1	Vorbereitungen.....	56
3.1.1	Szenario.....	56
3.1.2	Zielsysteme.....	57
3.1.2.1	Fujitsu Siemens Esprimo E5720.....	57
3.1.2.2	Microsoft Surface Pro 5 (Model 1807).....	58
3.1.2.3	Gigabyte H410M S2H V3 mit GC-TPM2.0 SPI 2.0.....	60
3.1.2.4	Gegenüberstellung.....	63
3.1.3	Betriebssysteme.....	64
3.1.4	Randbedingungen Logikanalysator.....	66
3.1.4.1	Amplituden-/ Frequenzgang messen.....	67
3.1.4.2	Bootdauer messen.....	71
3.1.5	Dekodier-Software (ARNE).....	72
3.1.5.1	main.py.....	73
3.1.5.2	var.py.....	76
3.1.5.3	lpc_decoder.py.....	76
3.1.5.4	spi_decoder.py.....	80
3.2	Experimentelle Ergebnisse.....	85
3.2.1	Verwendete Geräte.....	85

3.2.2	Vorgehen .....	87
3.2.3	Auswertung.....	100
4	Zusammenfassung.....	110
	Literaturverzeichnis .....	112
	Abbildungsverzeichnis.....	124
	Tabellenverzeichnis.....	127
	Abkürzungsverzeichnis .....	129
	Digitales Anlagenverzeichnis.....	133
	Selbständigkeitserklärung .....	141



# 1 Einleitung und Motivation

Die Zahl der vollverschlüsselten Systeme unter Privatanwendern nimmt immer weiter zu. Dies liegt daran, dass Hersteller von Verschlüsselungssystemen diese immer benutzerfreundlicher dem Endverbraucher anbieten können. Eine komfortable Möglichkeit der Festplattenverschlüsselung bietet dabei die Verwendung eines Trusted Platform Moduls (TPM). Der Schlüssel zum ver- und entschlüsseln der Daten wird durch ein ausgelagertes Modul geschützt. Dabei schützt es den Schlüssel der Festplattenverschlüsselung, ohne dass der Benutzer selbst, z.B. durch die Eingabe eines Passwortes, aktiv werden muss. Das TPM kann seit 2005 in Rechnersysteme verbaut werden, wobei es im Privatanwenderbereich bisher selten genutzt wurde. Mit der Einführung von Windows 11 wird sich dies voraussichtlich ändern. Dieses sieht in seinen Anforderungen das Vorhandensein eines TPM zwingend vor. Verschlüsselungssysteme, wie BitLocker, können dann das TPM verwenden. Zusätzlich hat Microsoft seine Politik bezüglich BitLocker geändert. Früher musste die Festplattenverschlüsselung aktiv vom Anwender eingerichtet werden. Seit neuestem wird BitLocker auf der Systempartition automatisch aktiviert, wenn bei der erst Einrichtung des Systems ein TPM und ein Microsoft Online Konto gegeben sind. Im Bereich Laptop und Desktop Endgeräte stellt Windows weiterhin den größten Marktanteil. Es ist deshalb anzunehmen, dass sich mit der Einführung von Windows 11, die Anzahl der Endgeräte mit TPM und damit auch die Anzahl von Systemen mit Festplattenverschlüsselung, basierend auf TPM, in Zukunft weiter erhöhen wird.

Für IT-Forensiker und Datenrettungsspezialisten sind Möglichkeiten der Entschlüsselung solcher Systeme von enormer Bedeutung. Es ist aktuell keine Forschungsarbeit bekannt, die sich mit der Funktionsweise und den verschiedenen Angriffen gegen TPM basierte Festplattenverschlüsselung auseinandersetzt. Mit Hilfe dieser Arbeit soll die Lücke nun geschlossen werden.



## **2 Theoretische Grundlagen**

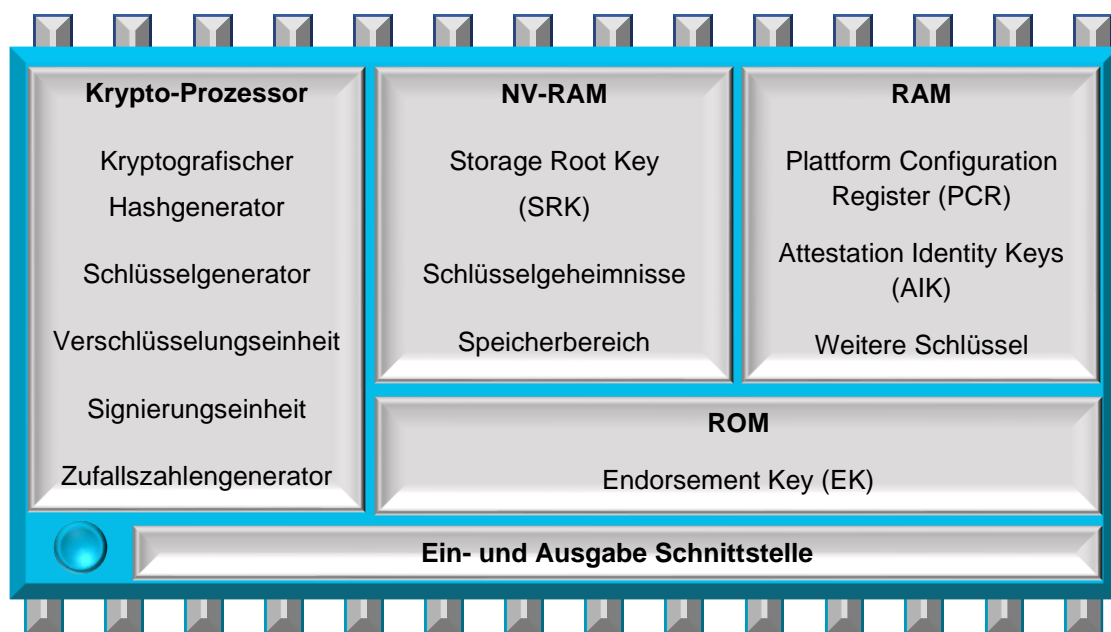
### **2.1 Trusted Platform Modul (TPM)**

Das Trusted Platform Modul (TPM) ist ein Hardware-Sicherheitsmodul. Es zählt zur selben Gruppe wie die Smartcard und das High-Level Security Modul (HLSM). Das TPM ist weit verbreitet, da es bei geringen Kosten eine hohe Sicherheit der Daten bietet. Anwendung findet es überwiegend in mobilen Computersystemen wie Notebooks. Grundsätzlich kann es jedoch in allen IT-Systemen wie z.B. PCs, Smartphones, Drucker, Router oder IoT-Geräte (TV, Waschmaschine etc.) zur Anwendung kommen. Aktuell befindet sich das TPM in der Version 2.0, dessen Eigenschaften und Funktionen sollen in diesem Abschnitt vorgestellt werden. [1, S. 101-106]

Das TPM ist ein wesentlicher Bestandteil der Trusted Computing Idee, welche die Integrität einer Ausführungsplattform sicherstellen soll. Es wurde von der Trusted Computing Group (TCG) im Jahr 2005 eingeführt. Die TCG entstand 2003 als Nachfolge Organisation der ursprünglich 1999 gegründeten Trusted Computing Platform Alliance (TCPA). Diese war ein Zusammenschluss der Unternehmen IBM, Intel, HP, AMD und Microsoft. [2, S. 234] Neben den bereits genannten Unternehmen gehören der TCG heute weitere zahlreiche Unternehmen an. Zu diesen gehören DELL, Google, HUAWEI und über 70 weitere Unternehmen aus dem Bereich der Hard- und Softwareentwicklung. [3] Das Konsortium hat sich selbst zum Ziel gesetzt, einheitliche Spezifikationen für Hard- und Software zu schaffen, die einen Betrieb in einer vertrauenswürdigen Rechnerumgebung ermöglichen. Auf der Webseite der TCG beschreibt sie sich selbst mit den Worten:

"The Trusted Computing Group (TCG) is a not-for-profit organization formed to develop, define and promote open, vendor-neutral, global industry specifications and standards, supportive of a hardware-based root of trust, for interoperable trusted computing platforms. TCG's core technologies include specifications and standards for the Trusted Platform Module (TPM), Trusted Network Communications (TNC) and network security and self-encrypting drives." [4]

### 2.1.1 Aufbau



**Abbildung 1:** Aufbau eines TPM (angelehnt an [2, S. 236])

Bei einem TPM handelt es sich in der Regel um einen integrierten Schaltkreis mit Recheneinheit, der über eigene Code- und Speicherbereiche verfügt. Innerhalb des Rechner-Systems stellt das TPM eine passive Komponente dar. Es kann über entsprechende Schnittstellen vom System angesprochen und verwendet werden. Der prinzipielle Aufbau eines TPM ist in **Abbildung 1** dargestellt. Es besteht im Wesentlichen aus einer E/A Schnittstelle, einem Krypto-Prozessor, einem flüchtigen (RAM) und einem nicht-flüchtigen (NV-RAM) Speicher, sowie einem Festwert-Speicher (ROM). [1, S. 104]

Die E/A Schnittstelle sorgt für die Kommunikation des Systems mit dem TPM und für die Zugriffsberechtigung. Nach Vorgabe der TCG kommuniziert das TPM über einen Low Pin Count- (LPC), einen Serial Peripheral Interface- (SPI) oder einen Inter-Integrated-Circuit- (I2C) Bus mit dem System. [5] Die Kommunikation über die beiden in der Praxis am häufigsten anzutreffenden Schnittstellen LPC und SPI werden in den Abschnitten 2.3.3.1 und 2.3.3.2 weiter erläutert.

Der Krypto-Prozessor stellt dem TPM erweiterte Sicherheitsfunktionen bereit. Zu diesen gehören: [6, S. 262]

- Kryptografische Hashfunktionen (SHA-1, SHA-256\*)
- Schlüsselgenerierung (RSA, ECC\*)
- Symmetrische und Asymmetrische Verschlüsselung (RSA)
- HMAC Berechnung
- Digitale Signaturen (RSA)
- Zufallszahlengenerator

\* Die markierten Funktionen stehen dem TPM erst seit der Spezifikation 2.0 zur Verfügung. [7]

Im flüchtigen Speicher (RAM) befindet sich das Platform Configuration Register (PCR), das die Ergebnisse der Integritätsmessung beinhaltet. Ebenfalls werden hier gespeicherte Schlüssel abgelegt.

Im nichtflüchtigen Speicher (NV-RAM) sind der Storage Root Key (SRK) und weitere Schlüsselgeheimnisse abgelegt.

Der Festwert-Speicher (ROM) beinhaltet den Endorsement Key (EK).

Die Bereiche des TPMs können zusammen mit der PC-Firmware zu den drei Komponenten Root of Trust for Reporting (RTR), Core Root of Trust for Measurement (CRTM) und dem Root of Trust for Storage (RTS) zusammengefasst werden.

- Der RTR und der CRTM dienen dazu, die Integrität eines Systems zu messen, abzuspeichern und zu vergleichen.
- Der RTS dient dazu den Zugriff auf Schlüssel, die im TPM gespeichert sind, zu verwalten.

Für das TPM ergeben sich somit die beiden Aufgaben der Integritätsmessung und der Schlüsselverwaltung. Diese sollen nachfolgend beschrieben werden. [2, S. 240]

### 2.1.2 Plattformintegrität und Integritätsmessung

Aus der Sicherheitsperspektive sollten Systeme immer in einer sicheren vertrauenswürdigen Umgebung ausgeführt werden. Eine vertrauenswürdige Umgebung zeichnet sich dadurch aus, dass es keine unerlaubten Veränderungen an der Hardware und dessen Firmware, sowie an der Software und anderen ausführbaren Codes gibt. Derartige Veränderungen führen zu einer Verletzung der Plattformintegrität und können dazu führen, dass das System nicht mehr ordnungsgemäß funktioniert oder angreifbar wird. Dem gegenüber stehen allerdings die gewollten Veränderungen des Systems, wie z.B. durch das Einspielen von Softwareupdates und der Austausch defekter oder veralteter Hardware. Für die Integritätsmessung ergeben sich somit zwei Aufgaben: [6, S. 261]

1. Sicherstellen der Plattformintegrität
2. Erkennen, an welcher Stelle Veränderungen stattgefunden haben

Das Grundprinzip der Integritätsmessung ist, den Zustand eines Systems initial bei der Einrichtung des TPMs zu messen und das Ergebnis zu speichern. Bei bestimmten Ereignissen, in der Regel dem Systemneustart (Power-On Reset), erfolgt eine erneute Integritätsmessung und die Werte werden mit den zuvor gespeicherten Werten verglichen. Die Integritätsmessung ist Teil des Trusted Computing und das TPM stellt diesem Prozess sein PCR zur Verfügung.

Um die Plattformintegrität sicherstellen zu können, werden messbare Objekte benötigt. Messbare Objekte können u.a. Hardwarekomponenten, Daten oder

digitale Signaturen sein. Welche dieser Objekte für die Integritätsmessung herangezogen werden, legt die Anwendung fest, die das TPM verwendet. Zur Integritätsmessung eines Objektes wird eine kryptografische Hashfunktion über das Objekt gebildet. Es wird ein eindeutiger Hashwert erzeugt, der den Zustand dieses Objektes repräsentiert. Ein Ergebnis der Integritätsmessung über ein Objekt wird in das PCR des TPM gespeichert. Das PCR kann mehrere Einträge fassen, so dass jede Messung einen eigenen Eintrag erhalten kann. Dies ermöglicht es zu erkennen, an welchem Objekt gegebenenfalls eine Veränderung stattgefunden hat. Es ist allerdings nicht zwingend notwendig, jedes gemessene Objekt mit einem eigenen Eintrag im PCR zu binden. Es können auch mehrere Messungen zusammen in dasselbe Register gesichert werden. Beispielsweise erhält nicht jede Einstellung der BIOS-Firmware einen eigenen Eintrag im PCR, sondern alle Einstellungen werden gemessen und gemeinsam in ein PCR geschrieben. Hierzu verkettet das TPM die Ergebnisse der Messungen aller Objekte mit Hilfe von Hashfunktionen. [6, S. 262-263]

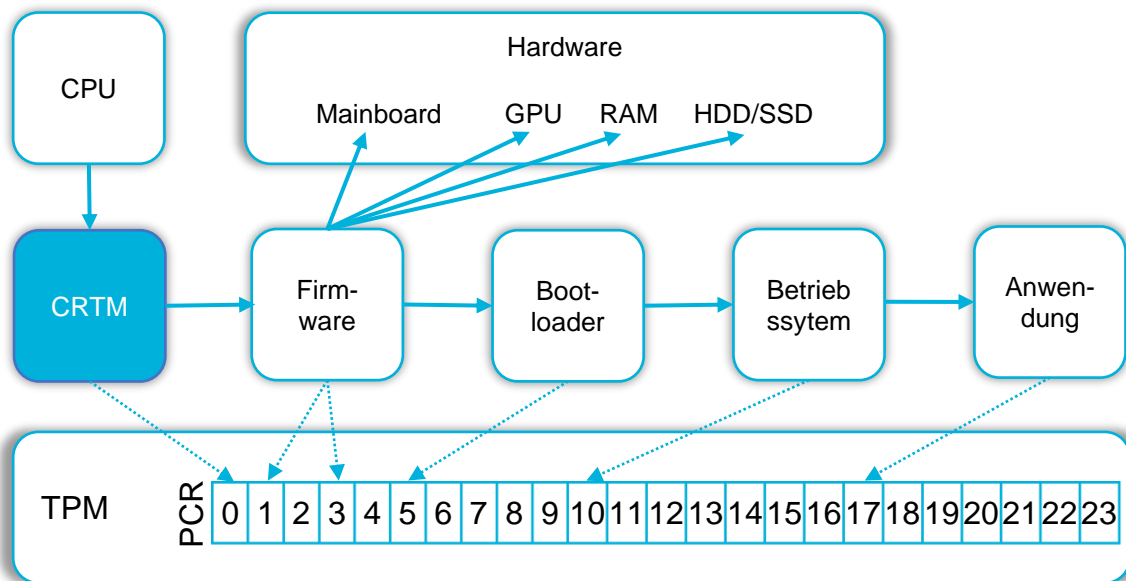
Nicht alle Register des PCRs können von der Anwendung und dem TPM frei belegt werden. Einige sind bereits für die Ergebnisse bestimmter Messung vorbelegt. In der **Tabelle 1** ist beispielhaft die PCR Belegung eines TPM in einem Computer dargestellt. Dabei muss unterschieden werden, in einer BIOS- oder UEFI-basierten Firmware. Es gilt zu beachten, dass die Ergebnisse der Integritätsmessung vom System nicht direkt in das PCR geschrieben werden können. Die Ergebnisse müssen zunächst an das TPM übertragen werden. Dieses beschreibt dann die PCRs. [8, S. 152]

Die **Abbildung 2** stellt exemplarisch den Ablauf einer Integritätsmessung bei Verwendung einer BIOS-Firmware dar. Diese verfolgt dabei die Idee der „Chain of Trust“. Hierbei soll das Vertrauen einer Komponente bewiesen werden, die sich am Ende einer Kette befindet. Dieses gilt nur dann als bewiesen, wenn allen vorherigen Komponenten ebenfalls vertraut wurde. [1, S. 251]

PCR	Zuweisung BIOS	Zuweisung EFI
0	CRTM, BIOS und Plattformerweiterungen	EFI unveränderlicher Startcode
1	BIOS-Einstellungen	EFI-Einstellungen
2	Option-ROM-Code	Veränderlicher EFI-Code
3	Option-ROM Konfiguration	Erweiterte oder austauschbare EFI-Einstellungen
4	MBR-Code	Boot-Manager
5	MBR-Partitionstabelle	GPT-Partitionstabelle
6	Statusübergangs- und Reaktivierungsereignisse	Statusübergangs- und Reaktivierungsereignisse
7	Plattformherstellerspezifische Messungen	Zustand Secure Boot
8	Betriebssystem Register	Reserviert
9	Betriebssystem Register	Reserviert
10	Betriebssystem Register	Reserviert
11	Betriebssystem Register	BitLocker Zugriffsteuerung
12	Betriebssystem Register	Daten und häufig eintretende Ereignisse
13	Betriebssystem Register	Boot Modul
14	Betriebssystem Register	Boot Berechtigungen
15	Betriebssystem Register	Reserviert
16	Debug	Reserviert
17-23	Anwendungsspezifisches Register	Reserviert

**Tabelle 1:** Vergleich PCR Register im BIOS- und UEFI-Firmware [8, S. 152]





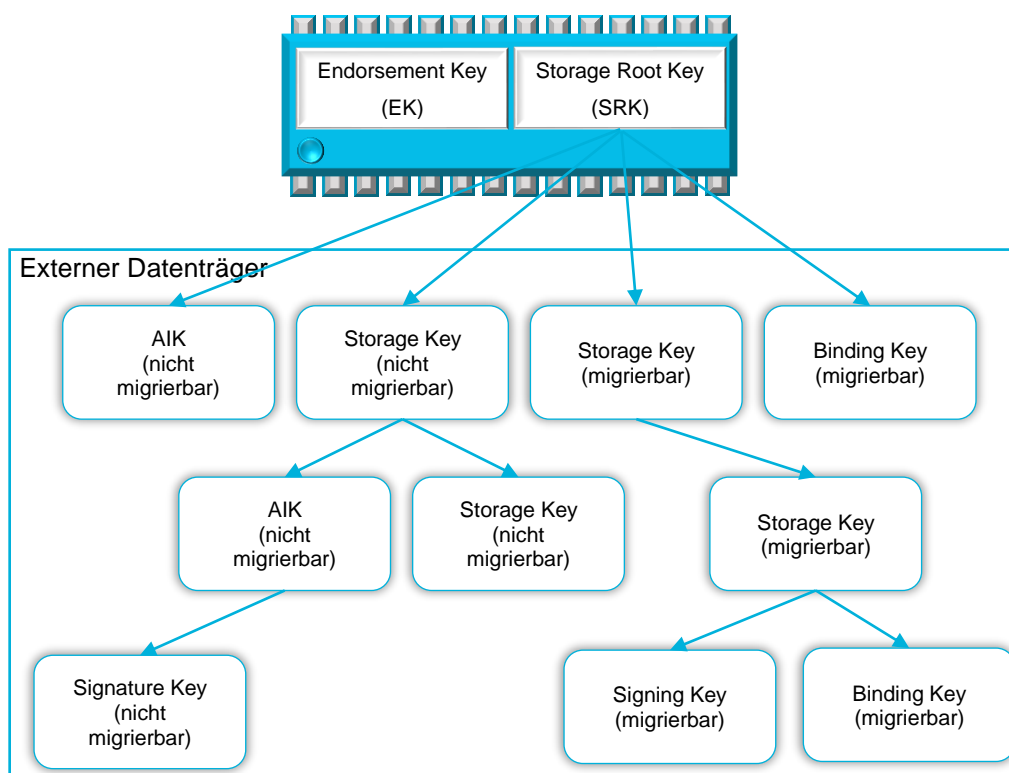
**Abbildung 2:** Integritätsmessung am Beispiel einer BIOS-Firmware (angelehnt an [6, S. 236])

Zum Start der Messung überschreibt das TPM alle PCR Einträge mit Nullen bzw. Einsen. [9, S. 151]. Anschließend folgt das Core Root of Trust Measurement (CRTM), das als Vertrauensanker im System angesehen werden kann. Dabei handelt es sich um einen unveränderlichen Code. Dieser wird von der Central Processing Unit (CPU) angewiesen, eine Integritätsmessung über die Firmware durchzuführen. Das CRTM ist der einzige Teil dieser Kette, über den selbst keine Integritätsmessung durchgeführt wird. Damit stellt es auch die einzige potenzielle Schwachstelle in der Kette dar. Es liegt an dem Plattformhersteller das CRTM zu schützen, indem er es unveränderlich macht. Dies erreicht er, indem er es in ein ROM speichert, Aktualisierungen des CRTM unterbindet oder nur Aktualisierungen mit signiertem Code zulässt. Das Ergebnis vom CRTM wird in das PCR[0] gesichert. Die Firmware wiederum führt eine Messung über die angeschlossene Hardware und den Bootloader durch. Die Ergebnisse dieser Messungen können in den Registern PCR[1] und folgende gesichert werden. Nachfolgend erfolgt vom Bootloader eine Integritätsmessung des Betriebssystems (PCR[8-13]), welches bei Bedarf dann die Integrität einer Anwendung misst. Dieses Ergebnis kann dann z.B. in PCR[14] gesichert werden.

So kann die Integrität jeder einzelnen Komponenten Schritt für Schritt nachgewiesen werden. Die Funktion - Veränderungen an der Plattform zu erkennen - wird als Authenticated Boot (auch als Trusted Boot) bezeichnet. Der Authenticated Boot zeichnet sich dadurch aus, dass lediglich die Veränderung erkannt wird. Das angemessene Reagieren auf diese Veränderung obliegt der Anwendung, die auf das TPM zurückgreift. Hier unterscheidet sich Authenticated Boot von z.B. Secure Boot, das beim Auftreten von Fehlern direkt den Bootvorgang beendet. [1, S. 250]

### 2.1.3 Schlüsselverwaltung

Eine weitere Anforderung an das TPM ist die der Schlüsselerzeugung und -verwaltung. Hierzu verwendet das TPM eine Schlüsselhierarchie, die so konzipiert ist, dass die Schlüssel aufeinander aufbauen. Ein Schlüssel ist immer durch den ihm übergeordneten Schlüssel geschützt. In **Abbildung 3** ist diese Schlüsselhierarchie exemplarisch dargestellt.



**Abbildung 3:** TPM Schlüsselhierarchie

Es wird unterschieden in Schlüsseln, die sich im TPM befinden und denen, die auf einem externen Datenträger gespeichert sein können. Da die Speichergröße des TPM begrenzt ist, werden mit Ausnahme des Endorsement Key (EK) und des Storage Root Key (SRK) die meisten Schlüssel ausgelagert. Auf einem externen Datenträger kann so die Schlüsselhierarchie beliebig fortgesetzt werden und ist nur von der Größe des externen Speichers begrenzt. Des Weiteren unterscheidet man bei den Schlüsseln zwischen migrierbaren und nicht migrierbaren Schlüsseln. Migrierbare Schlüssel können zwischen Plattformen getauscht werden, sodass diese auch von anderen TPMs verwendet werden können. Das Migrieren von Schlüsseln erweitert zwar deren Anwendungsbereich, führt aber auch dazu, dass sie nicht mehr eindeutig einem TPM zugeordnet werden können. Die nicht migrierbaren Schlüssel können zwar nicht mit anderen Plattformen getauscht werden, dafür ist sichergestellt, dass nur das TPM, welches den Schlüssel erzeugt hat, ihn benutzen darf. Ob ein Schlüssel migrierbar oder nicht migrierbar ist, wird durch Zertifikate, die bei der Schlüsselerzeugung erstellt werden, festgelegt. [1, S. 252-253]

Jedes TPM verfügt über den Endorsement Key (EK), der kein Teil der Schlüsselhierarchie ist. Die Aufgabe dieses Schlüssels ist es, die Identität des TPM sicherzustellen. Es handelt sich um ein 2048-Bit langes RSA Schlüsselpaar. Erzeugt wird das Schlüsselpaar von einer Zertifizierungsstelle (z.B. dem TPM Hersteller). Um die Identität sicherzustellen, kann der öffentliche Teil des Schlüsselpaares abgefragt werden, während der private Teil immer im TPM verbleibt und nicht verändert werden kann. [10, S. 31]

Den Ausgangspunkt der Schlüsselhierarchie wiederum stellt der Storage Root Key (SRK) dar. Dieser befindet sich im NV-RAM Bereich des TPM. Wenn eine Anwendung zum ersten Mal das TPM verwenden möchte, weist diese das TPM an, einen 2048-Bit langes RSA Schlüsselpaar zu erzeugen. Mit Hilfe des SRK kann dann der darunter liegende Schlüssel, der nicht mehr im TPM liegen muss, verschlüsselt werden. Der SRK verlässt das TPM niemals, kann aber bei Bedarf gelöscht oder erneuert werden. Wenn dies geschieht, hat dies allerdings zur Folge, dass die darunter liegenden Schlüssel nicht mehr entschlüsselt werden können. Der SRK als Ausgangsschlüssel eröffnet somit die Möglichkeit, dass

andere Schlüssel kryptografisch sicher auf externen Datenträgern gespeichert werden können. [1, S. 253]

Neben diesen beiden Schlüsseln im TPM existieren noch folgende weitere Schlüssel:

- Attestation Identity Key (AIK), nicht migrierbarer RSA-Schlüssel, der u.a. zum Signieren der Werte genutzt wird, welche in das PCR geschrieben werden sollen. Der AIK kann extern gespeichert werden, muss dann aber durch Verschlüsselung geschützt werden. [10, S. 61]
- Storage Key, dient zur Verschlüsselung von Daten auf einem externen Medium und kann als nicht migrierbar und migrierbar ausgeführt sein. [1, S. 254]
- Binding Key, Benutzer Schlüssel werden durch den Binding Key verschlüsselt und können so auch auf anderen Plattformen sicher verwendet werden. Binding Keys sind migrierbar. [1, S. 254]
- Signing Key, funktionieren wie Binding Keys mit der Besonderheit, dass bei der Verschlüsselung die Plattformintegrität mitberücksichtigt wird. Signing Keys sind migrierbar. [1, S. 254]
- Signature Key, Schlüssel zur Erzeugung von digitalen Signaturen. Wird ein Signature Key von einem übergeordneten AIK geschützt, benötigt der Schlüssel kein eigenes Zertifikat, da das Zurückführen eines Signatures Keys auf ein den dazugehörigen AIK für die Sicherstellung der Signatur ausreicht. [2, S. 237]

Mit Hilfe der Schlüsselerzeugung und -verwaltung ergeben sich für das TPM die beiden Sicherheitsfunktionen der Auslagerung (binding/wrapping) und des Ver- und Entsiegeln (Sealing). Bei der Auslagerung werden Benutzergeheimnisse wie z.B. geheime Schlüssel durch einen Binding Key verschlüsselt. Da der Binding Key migrierbar ist, kann der verschlüsselte geheime Schlüssel auch auf anderen Plattformen mit dem entsprechenden Binding Key entschlüsselt werden. Das Sealing funktioniert identisch dem Binding, mit der Besonderheit, dass der geheime Schlüssel mit dem SRK des TPM verschlüsselt wird. Dadurch ist der geheime Schlüssel an das TPM gebunden und kann nur von diesen entschlüsselt

werden. Im Zusammenspiel mit der Integritätsmessung kann das TPM dafür sorgen, dass es nur bei erfolgreich verifizierter Plattformintegrität den geheimen Schlüssel entsiegelt. [1, S. 250,257]

## **2.1.4 Schutzmaßnahmen**

Das TPM verfügt über integrierte Schutzmaßnahmen, die es vor Angriffen schützen soll. Zu diesen gehören u.a. die Reset Attack Mitigation, die Parameter Encryption und die Anti-Hammering Funktion.

### **2.1.4.1 Reset Attack Mitigation**

Der Schlüssel einer Festplattenverschlüsselung wird, nach erfolgreicher Entsiegelung durch den SRK, in der Regel in den Arbeitsspeicher des Systems geladen. Es besteht das Problem, dass ein Angreifer den Entsiegelungsprozess des TPM umgeht, in dem er nach dem Entsiegeln den Arbeitsspeicher ausliest. Dies könnte er beispielsweise durch einen Neustart des Systems mit einem angepassten Betriebssystem tun. Um dies zu unterbinden, wird beim Einrichten des TPM das Memory Overwrite Request (MOR) Bit in der Firmware des Systems gesetzt. Dieses MOR-Bit weist, je nach Zustand, das System dazu an den Arbeitsspeicher beim Start zu überschreiben. Dadurch werden alle Schlüssel und Daten aus dem Arbeitsspeicher gelöscht. Das MOR-Bit wird nach erfolgreicher Firmware Initialisierung und vor dem Start des Betriebssystems auf den Wert „Eins“ gesetzt. Wird das Betriebssystem ordnungsgemäß heruntergefahren, so werden der Schlüssel aus dem Arbeitsspeicher gelöscht und das MOR-Bit auf den Wert „Null“ zurückgesetzt. Beim nächsten Start des Systems erfolgt keine Überschreibung des Arbeitsspeichers. Wird das System über einen Reset im laufenden Betrieb neugestartet, so besitzt das MOR-Bit weiterhin den Wert „Eins“ und beim nächsten Start wird der Arbeitsspeicher überschrieben. [11]

#### 2.1.4.2 Parameter Encryption

Mit der Spezifikation TPM2.0 wurde die Parameter Encryption eingeführt. Diese erlaubt es Befehle mit verschlüsselten Parametern an das TPM zu senden. So ist es z.B. möglich, den Befehl zum Entsiegeln eines Schlüssels mit dem SRK an das TPM zu senden, wobei der Parameter, der den versiegelten Schlüssel enthält, verschlüsselt wird. Andersrum kann das TPM ebenfalls Befehle mit verschlüsselten Parametern an den Benutzer zurückgeben.

Die verschlüsselte Kommunikation ermöglicht eine abhörsichere Kommunikation zwischen den beiden Kommunikationspartnern.

Um die Parameter Encryption nutzen zu können, müssen sich beide Kommunikationspartner zuvor auf eine verschlüsselte Kommunikation einigen. [12]

#### 2.1.4.3 Anti-Hammering

Befehle wie das Erstellen eines SRK und das Ver- und Entsiegeln von Schlüsseln mit dem SRK kann in der Regel nur der Besitzer des TPMs ausführen. Dieser wird mit der ersten Initialisierung des TPM festgelegt. Um zu verhindern, dass auch andere Benutzer Befehle an das TPM senden können, verfügt der TPM-Besitzer über einen geheimen Schlüssel, die AuthData. Mit diesem weist sich der Nutzer dem TPM gegenüber als TPM-Besitzer aus. Befehle an das TPM werden gemeinsam mit den AuthData übermittelt. [13, S. 65]

Die AuthData werden sicher im System des TPM-Besitzers verwaltet und können von außen nicht ausgelesen werden.

Es besteht die Gefahr, dass ein Angreifer versucht, mittels Brute-Force Befehle an das TPM zu senden. Beispielsweise um einen verschlüsselten Schlüssel mit dem SRK entschlüsseln zu lassen. Hierzu sendet der Angreifer mehrere Befehle mit dem Schlüssel und generierten AuthData, bis das TPM den Befehl zum Entschlüsseln ausführt.

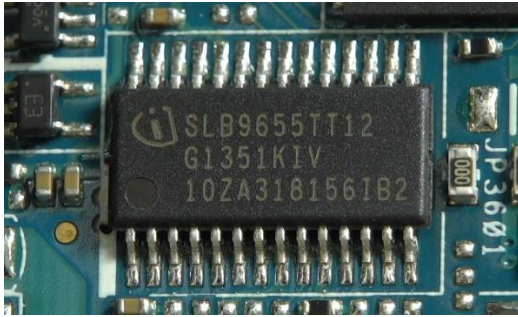
Um dies zu unterbinden, ist im TPM eine Anti-Hammering Funktion implementiert. Diese zählt die Anzahl fehlgeschlagener Autorisierungen und sperrt das TPM nach 32 fehlgeschlagenen Versuchen. Wurde für 10 Minuten keine neue fehlgeschlagene Autorisierung festgestellt, sinkt der Zähler um den Wert eins und es kann ein neuer Versuch gestartet werden. Der Zähler sinkt nur, wenn das System eingeschaltet ist oder sich im Energiesparmodus befindet. [14]

### 2.1.5 Ausführungsvarianten

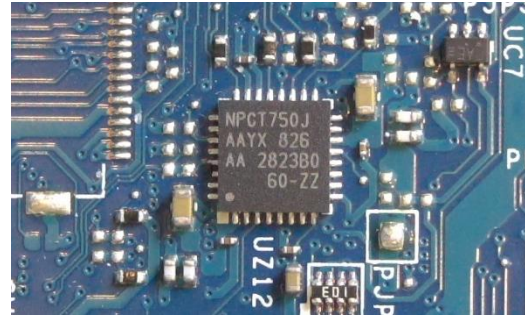
Bei einem TPM kann zwischen einem diskreten TPM (dTPM) und einem firmware TPM (fTPM) unterschieden werden.

Das fTPM ist eine Implementierung der TPM-Sicherheitsfunktionen innerhalb der CPU-Architektur. Intel nennt diese Funktion Intel Platform Trust Technology (PTT) und bei AMD wird es als AMD Secure Prozessor bezeichnet. Im Unterschied zu einem dTPM verfügt das fTPM u.a. über keinen echten Zufallszahlengenerator, der für die Erzeugung der Schlüssel benötigt wird. Dieser wird bei einem fTPM durch Softwarelösungen ersetzt, die aber auf Algorithmen beruhen und somit keine echten Zufallszahlen mehr liefern. Ein weiteres Problem des fTPM ist es, sichere Ressourcen außerhalb der CPU für das Trusted Computing bereitzustellen. Auf dem 25. USENIX Security Symposium wurde von mehreren Wissenschaftlern die Arbeit „*fTPM: A Software-Only Implementation of a TPM Chip*“ vorgestellt, die sich mit den Herausforderungen des fTPM befasst. [15]

Der Schwerpunkt dieser Masterarbeit soll auf dem dTPM liegen, welches über einen eigenen Chip mit Recheneinheit auf der Plattform realisiert wird. Mit den Spezifikationen der TCG existieren für das dTPM Empfehlungen für das Package und Pinout Design. Die meisten Hersteller folgen diesen Empfehlungen, da so Kosten für ein eigenes Design eingespart werden können. TPMs sind meist in einem 28-Pin TSSOP bzw. einem 32-Pin QFN Package untergebracht. [5, S. 167] In der **Abbildung 4** und **Abbildung 5** sind beide Bauformen exemplarisch dargestellt.



**Abbildung 4:** 28-Pin TSSOP



**Abbildung 5:** 32-Pin QFN

Diese Chips sind entweder fest auf dem Mainboard verlötet oder können, wie bei den meisten Desktop-PCs, als Header Modul aufgesteckt werden. Hersteller solcher dTPM-Chips sind u.a. Infineon, Nuvoton und STMicroelectronics.

### 2.1.6 Anwendungen

Das TPM wird mittlerweile von mehreren Sicherheits-Anwendungen unterstützt. Zu diesen zählen u.a. DM-Crypt, Trusted Boot, Windows Hello-PIN und BitLocker. Die Festplattenverschlüsselung BitLocker soll im nächsten Kapitel vorgestellt werden.

## 2.2 Festplattenverschlüsselungen

Festplattenverschlüsselungen dienen dem Schutz vor unberechtigtem Zugriff dritter auf die eigenen Daten. Hierzu verschlüsselt sie die gespeicherten Daten auf dem Speichermedium so, dass sie nicht mehr lesbar sind. Nur durch entsprechende Entschlüsselung können die Daten lesbar gemacht werden.

Es existieren Festplattenverschlüsselungen in Software- und Hardware-basierten Ausführungen, wobei meist eine symmetrische Verschlüsselung verwendet wird. Merkmal der symmetrischen Verschlüsselung ist es, dass die Daten mit demselben Schlüssel ver- und entschlüsselt werden können. Nur der Besitzer des Schlüssels kann die Daten wieder zu lesbaren Inhalten umwandeln. Ein entscheidendes Kriterium bei der Festplattenverschlüsselung ist es, dass sich weder die Datenmenge noch die Anzahl der Daten, die gelesen und geschrieben werden müssen, signifikant erhöhen darf, da der Speicher auf einer



Festplatte begrenzt ist. Erreicht werden kann dies durch die geeignete Wahl von Blockchiffrierung. [16, S. 24-25]

### **2.2.1 Vergleich Softwarebasierter Festplattenverschlüsselung**

Es existieren unterschiedliche Ansätze der Festplattenverschlüsselungen, bei den entweder die gesamte Festplatte, Laufwerkspartitionen oder Bereiche innerhalb der Partition verschlüsselt werden. Nachfolgend sollen die vier softwarebasierten Festplattenverschlüsselungen BitLocker, FileVault, LUKS und VeraCrypt vorgestellt und in einer Tabelle miteinander verglichen werden.

#### BitLocker

BitLocker ist eine seit Windows Vista verfügbare Festplattenverschlüsselung von Microsoft. Sie ist bei den Versionen Windows Vista und Windows 7 nur in den Editionen Ultimate und Enterprise bzw. ab Windows 8 in den Editionen Pro und Enterprise enthalten. BitLocker verwendet in der neuesten Version eine XTS-AES Verschlüsselung mit einer Schlüssellänge von 128-Bit oder 256-Bit. Mit BitLocker können Systempartitionen, Partitionen und externe Speichermedien verschlüsselt werden. Zum Ver- und Entschlüsseln verfügt BitLocker über eine komplexe Schlüsselhierarchie. Mit dem Full Volume Encryption Key (FVEK) werden die Daten ver- bzw. entschlüsselt. Der FVEK wird mit dem Volume Master Key (VMK) verschlüsselt und der VMK liegt in verschlüsselter Form auf der verschlüsselten Partition. Der VMK muss vom Benutzer entschlüsselt werden, wofür BitLocker verschiedene Authentifizierungsmethoden bereitstellt. So kann dieser Schlüssel mit Hilfe eines Passwortes, einer Smartcard, eines TPM, eines Geheimnisses auf einem USB-Stick oder mit dem Wiederherstellungsschlüssel entschlüsselt werden. [17] Microsoft empfiehlt BitLocker in Kombination mit einem TPM für größtmöglichen Schutz zu verwenden. [18] Erfüllt die Plattform alle Bedingungen für BitLocker und wird bei der erstmaligen Einrichtung des Systems ein Microsoft Online Konto angegeben, so wird BitLocker auf dem System automatisch aktiviert. [19]

## FileVault

FileVault ist die Festplattenverschlüsselung von Apple MacOS. Sie wurde mit der Version OS X Panther (10.3) eingeführt. Seit dem Betriebssystem OS X Lion (10.7) wurde FileVault durch FileVault 2 abgelöst, auf das sich nachfolgend bezogen wird. FileVault ist eine Volume Verschlüsselung basierend auf dem Core Storage. Für die Verschlüsselung wird der XTS-AES 128-bit Algorithmus mit einer Schlüssellänge von 256-Bit verwendet. Mit FileVault können sowohl interne als auch externe Speichermedien verschlüsselt werden. [20] [21] Mit der Einführung des Apple T2 Chips, hat sich das Anwendungsgebiet von FileVault als Festplattenverschlüsselung um den Punkt der Zugriffsberechtigung erweitert. Die Dateien auf solchen Geräten werden nun mit dem T2 Chip verschlüsselt, während FileVault diesen mit einem Passwort schützt. Erst nach erfolgreicher Eingabe des Passwortes durch den Benutzer beginnt der T2 Chip mit der Entschlüsselung des Systems. [22]

## LUKS (DM-Crypt)

LUKS ist eine Erweiterung des Kernel Moduls DM-Crypt, welche sich auf Linux Betriebssystemen durchgesetzt hat. Mit LUKS können ganze Festplatten oder einzelne Partitionen verschlüsselt werden. Die Verschlüsselung findet auf Block Device Ebene statt. Nach erfolgreicher Entsperrung durch den Benutzer wird das Gerät als virtuelles Device bereitgestellt. Der Zugriff auf das virtuelle Device wird durch DM-Tabellen gesteuert. Damit der Benutzer auf die DM-Tabellen zugreifen kann, muss dieser das Passwort kennen. Zur Schlüsselgenerierung und Verschlüsselung des Devices dient das Tool cryptsetup. Dieses stellt unterschiedlichste Modi und Algorithmen zur Verschlüsselung von Geräten bereit. Die Schlüsselerzeugung erfolgt über die Password-Based Key Derivation Function 2 (PBKDF2). LUKS bietet einen einheitlichen Header für das verschlüsselte Gerät. Dieser beinhaltet u.a. Informationen zur vorliegenden Verschlüsselung, damit der Benutzer diese bei der Entschlüsselung nicht mit angeben muss. Eine Besonderheit von LUKS ist es, dass bis zu acht unterschiedlichen Passwörtern gespeichert werden. Diese können zum Entsperren des Gerätes verwendet werden. Dadurch ist es möglich, dass

mehrere Benutzer Zugriff auf ein Gerät erhalten, ohne das Passwort des jeweils anderen zu kennen. [23] LUKS kann zum Ver- und Entschlüsseln der Devices das TPM verwenden. Diese Funktion befindet sich aktuell in der experimentellen Phase. [24]

### VeraCrypt

Als einziges der hier vorgestellten Programme bietet VeraCrypt die Möglichkeit der plattformübergreifenden Verschlüsselung. Es kann unter Windows, Linux und MacOS verwendet werden. VeraCrypt ist der inoffizielle Nachfolger des TrueCrypt Programmes, welches im Mai 2014 eingestellt wurde. Es basiert auf der TrueCrypt Version 7.1a und bietet weiterhin die Möglichkeit TrueCrypt Verschlüsselungen zu entschlüsseln. Mit VeraCrypt ist es möglich, verschlüsselte Partitionen, versteckte verschlüsselte Partitionen, verschlüsselte Container und verschlüsselte Systempartitionen anzulegen. Das Erzeugen von verschlüsselten Systempartitionen ist allerdings nur dem Windows Betriebssystem vorbehalten. Für die Verschlüsselung bietet VeraCrypt verschiedene Verschlüsselungs-Algorithmen. Eine besondere Stärke von VeraCrypt ist die Kaskadierung mehrerer Verschlüsselungsalgorithmen, die ein Entschlüsseln erschwert. Zusätzlich zur Kaskadierung hat VeraCrypt die Funktion des Personal Iterations Multiplier (PIM) implementiert. Diese erlaubt es dem Benutzer die Anzahl der Verschlüsselungs-Iterationen selbst festzulegen. Dies führt dazu, dass bei einem Brute-Force Angriff verschiedene Passwörter, Verschlüsselungsalgorithmen und Iterations durchprobiert werden müssen. Die Erzeugung des Passwortes erfolgt über die PBKDF2 Funktion. [25]

In der **Tabelle 2** sind noch einmal alle hier vorgestellten softwarebasierten Festplattenverschlüsselung gegenübergestellt. Dabei sind deren Blockchiffre, die möglichen Verschlüsselungsalgorithmen und Besonderheiten aufgeführt.

Software	OS	Block- chiffre	Algorithmus	Besonderheit
Bitlocker	Windows	CBC XTS	AES	TPM
FileVault	MacOS	XTS	AES	T2
LUKS	Linux	ECB	AES	TPM
		CBC	SERPENT	(experimentell)
		XTS	TWOFISH	Bis zu 8 Kennwörter
VeraCrypt	Linux	XTS	AES	PIM
	MacOS		SERPENT	Kaskadierung
	Windows		TWOFISH	
			CAMELLIA KUZNYECHIK	

**Table 2:** Vergleich Verschlüsselungssoftware

Allen hier vorgestellten Verschlüsselungsprogrammen gemeinsam ist die XTS-AES Verschlüsselung. Diese wurde von dem National Institute of Standards and Technology (NIST) als sicher eingestuft. [26] Ein Brechen des Verschlüsselungsalgorithmus gilt aus heutiger Sicht als nicht möglich. Ein erfolgreicher Angriff gegen diese Verschlüsselung richtet sich deshalb immer gegen den symmetrischen Schlüssel. Von den vorgestellten Programmen bietet nur BitLocker eine voll funktionsfähige Verschlüsselung, welche das TPM nutzt. Nachfolgend soll deshalb der BitLocker Algorithmus vertiefend untersucht werden.

### 2.2.2 Der BitLocker Algorithmus

Dabei wird zunächst der BitLocker Entschlüsselungsprozess einer Systempartition analysiert. Anhand dieser sollen Grundlagen der BitLocker Schlüssel, der Schlüsselhierarchie und die Schutzmaßnahmen dieser Schlüssel

beschrieben werden. Mit Hilfe der Analyse können mögliche Schwachstellen der BitLocker Verschlüsselungen mit TPM identifiziert werden. Es wird sich hierbei ausschließlich auf die BitLocker Systemverschlüsselung bezogen. Das Verschlüsseln von Wechseldatenträgern mit BitLocker To Go wird nicht weiter betrachtet, da dieses kein TPM als Schutzmaßnahme verwendet.

BitLocker verwendet mehrere Schlüssel, die aufeinander aufbauen und den jeweils darunter liegenden Schlüssel verschlüsseln. Am Ende dieser Kette befindet sich der Full Volume Encryption Key (FVEK), mit dem die Rohdaten auf der Partition ver- bzw. entschlüsselt werden. Die **Abbildung 6** beschreibt den vollständigen Ablauf, um vom ersten Schlüssel, dem versiegelten Volume Master Key (VMK), bis zum FVEK zu gelangen.

Der Prozess beginnt bei der BitLocker-Partition und dem Extrahieren des Volume Master Keys (VMK) in versiegelter Form.

#### Volume Master Key (VMK), versiegelt:

Der versiegelte VMK ist auf der verschlüsselten BitLocker-Partition abgelegt und muss beim Starten des Systems zunächst entsiegelt werden. Dieser befindet sich innerhalb eines FVE-Metadateneintrages. Jede BitLocker-Partition enthält drei FVE-Metadatenblöcke, welche mit der Signatur '-FVE-FS-' starten. Innerhalb eines Metadatenblockes können mehrere FVE-Metadateneinträge enthalten sein. In solch einem FVE-Metadateneintrag befindet sich der versiegelte VMK in mehrfacher Ausführung. BitLocker speichert die Informationen zum versiegelten VMK redundant, sodass bei Beschädigung eines Eintrages ein weiterer zur Verfügung steht. Der VMK kann durch unterschiedliche Schutzmaßnahmen von BitLocker mit einem AES- bzw. RSA-Algorithmus versiegelt werden. In der Regel werden mindestens zwei versiegelte VMK auf der BitLocker-Partition abgelegt, die mit unterschiedlichen Schlüsseln entsiegelt werden können. Um mit dem Prozess erfolgreich fortzufahren, genügt es allerdings nur einen der VMKs zu entsiegeln, da alle zum selben FVEK führen. [27, S. 19]

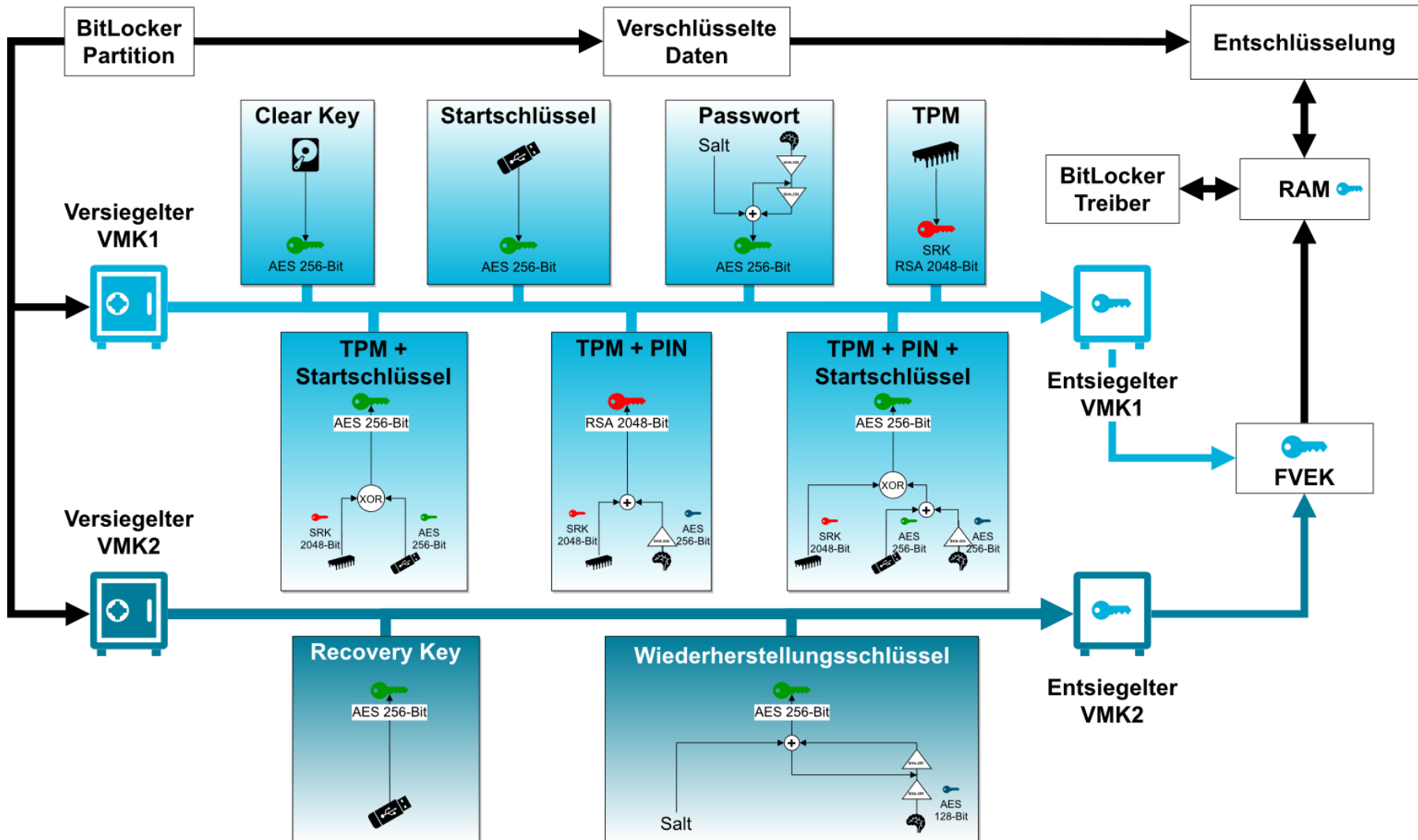


Abbildung 6: BitLocker Systemstart

Um den versiegelten VMK zu entsiegeln, durchläuft BitLocker eine festgelegte Reihenfolge, bei der das Entsiegeln mit den unterschiedlichen Authentifizierungsmaßnahmen ausprobiert wird. War das Entsiegeln durch eine Maßnahme erfolglos, wird mit der nächsten fortgefahren, bis alle Maßnahmen getestet wurden. Nur wenn keine Authentifizierungsmaßnahme erfolgreich war, bricht der BitLocker Prozess ab. Die erste Maßnahme, die durchlaufen wird, ist der Clear Key. [28, S. 10-11]

### Clear Key

Der Clear Key ist streng genommen keine echte Authentifizierungsmaßnahme. Bei dem Clear Key handelt es sich um einen 256-Bit AES Schlüssel, welcher im Klartext auf der BitLocker-Partition mit dem versiegelten VMK abgelegt ist. Mit diesem kann der VMK entsiegelt werden, sodass bei physikalischem Verlust des Datenträgers kein Schutz mehr gegeben ist. Anwendung findet der Clear Key, wenn beispielsweise ein Administrator über Fernwartung das System starten muss, ohne vor Ort eine Authentifizierung durchführen zu können. [29]

Schlägt die Authentifizierung mittels Clear Key fehl, überprüft BitLocker das Vorhandensein eines Startschlüssels.

### Startschlüssel (USB)

Eine komfortable Variante des Authentifizierungsmechanismus stellt der Startschlüssel da. Bei diesem handelt es sich um eine Schlüssel-Datei mit der Endung \*.BEK, mit welcher der VMK über den enthaltenen AES-Schlüssel entsiegelt wird. Die Datei wird auf einem externen Datenträger, wie z.B. einem USB-Stick versteckt, abgelegt. Zum Systemstart muss der Datenträger angeschlossen sein, damit BitLocker von diesem den Schlüssel laden kann. Konnte der Schlüssel gefunden werden, ist keine weitere Interaktion des Benutzers notwendig. [29]

Konnte kein Startschlüssel gefunden werden, prüft BitLocker das Vorhandensein eines TPM auf dem System. Konnte ein TPM gefunden werden, werden die nachfolgenden TPM-Schutzvarianten getestet. Befindet sich kein TPM auf dem

System, werden diese Maßnahmen übersprungen und mit der Abfrage eines Passwortes fortgefahren.

### TPM

Die zuvor genannten Mechanismen schützen zwar den VMK, stellen aber nicht die Plattformintegrität sicher. Ein Angreifer, der physikalischen Zugriff zum Datenträger hat, könnte das System verändern, um Angriffe gegen dieses durchzuführen. Die Empfehlung zum Schutz des VMK von Microsoft lautet daher: „BitLocker bietet den größten Schutz bei Verwendung mit einem Trusted Platform Module (TPM)“. [18] Ist ein TPM vorhanden, wählt BitLocker diesen Schutz als Standard. Beim Starten des Systems wird die Plattformintegrität überprüft. Der versiegelte VMK wird an das TPM gesendet und dort mit dem SRK entsiegelt. Der VMK ist bei diesem Schutzmechanismus mit einem RSA-Verschlüsselungsalgorithmus geschützt. Das Entsiegeln läuft im Hintergrund, sodass keine weitere Benutzerinteraktion beim Systemstart notwendig ist. [30]

### TPM mit PIN

Die Schutzvariante mit einem TPM kann um die Sicherheitskomponente eines PIN erweitert werden. Bei der PIN handelt es sich um eine Zeichenkette (im Standard ausschließlich numerische Zeichen), die der Benutzer beim Systemstart angeben muss. Der versiegelte VMK und die PIN werden an das TPM gesendet. Der Schlüssel zum Entsiegeln des VMK setzt sich aus dem SRK und aus der PIN zusammen, wobei zunächst aus der PIN ein SHA-256 Hash gebildet wird. Nur wenn der SRK freigegeben wurde und der Hashwert aus der PIN korrekt war, kann der VMK entsiegelt werden. Der VMK ist bei dieser Variante ebenfalls mit einem RSA-Verschlüsselungsalgorithmus geschützt. [30]

### TPM mit Startschlüssel (USB)

Anstatt eine PIN zu setzen, besteht die Möglichkeit der Kombination aus SRK und einem Startschlüssel. Der Startschlüssel befindet sich wieder auf einem Wechseldatenträger und muss beim Start des Systems mit diesem verbunden sein. Über eine XOR-Verknüpfung aus SRK und Startschlüssel wird der VMK



entsiegelt. Der VMK ist mit einem AES-Verschlüsselungsalgorithmus geschützt. [30]

### TPM mit PIN und Startschlüssel (USB)

Den größtmöglichen Schutz bietet BitLocker bei der Verwendung eines TPM in Kombination aus PIN und Startschlüssel. Ein Angreifer müsste so in den physikalischen Besitz des Systems und des Wechseldatenträgers gelangen. Zusätzlich benötigt er dann immer noch das Wissen über die vom Benutzer gesetzte PIN. Aus dem Startschlüssel und dem SHA-256 Hash, welcher aus der PIN gebildet wurde, wird ein neuer Schlüssel gebildet. Aus diesem Schlüssel und dem SRK wird über eine XOR Verknüpfung der AES Schlüssel zum Entschlüsseln des versiegelten VMK gebildet. [30]

### Passwort

Das Setzen eines Passwortes stellt den einfachsten Authentifizierungsmechanismus zum Schutz des VMK da. Der Benutzer legt bei der Einrichtung von BitLocker das Passwort fest. Bei jedem Start des Systems wird der Benutzer nach diesem Passwort gefragt. Das Passwort ergibt gemeinsam mit einem von BitLocker festgelegten Salt einen 256-Bit AES Schlüssel, mit dem der VMK entsiegelt werden kann. Erst wenn der VMK mit dem Passwort erfolgreich entsiegelt wurde, startet das Betriebssystem. Diese Variante des Authentifizierens ist Standard bei der Verschlüsselung von Wechseldatenträgern. [29]

Das Problem bei den oben vorgestellten Schutzmechanismen ist, dass bei Verlust bzw. Vergessen des Passwortes, USB-Stick, SRK oder der PIN die Daten nicht mehr entschlüsselt werden können. Im schlimmsten Fall sind die Daten dann verloren. Um dies zu vermeiden, erzeugt BitLocker bei der Einrichtung einen zweiten VMK, der über einen alternativen AES-Schlüssel entschlüsselt werden kann. Dabei handelt es sich um BitLocker Wiederherstellungsmaßnahmen, die hier kurz vorgestellt werden sollen.

Recovery Key (USB)

Zur Wiederherstellung kann ein Recovery Key erzeugt und auf einem Wechseldatenträger gespeichert werden. Das Prinzip ist gleich dem Startschlüssel, bei dem eine \*.BEK auf dem Wechseldatenträger abgelegt wird.

Wurde kein Recovery Key gefunden, besteht noch die Möglichkeit der Abfrage des Wiederherstellungsschlüssels. [29]

Wiederstellungsschlüssel

Bei dem Wiederherstellungsschlüssel handelt es sich um eine 48-stellige Zahlenkombination, welche als Passwort dient. Die Zahlen werden zu acht Zahlenblöcke aus jeweils sechs Zahlen zusammengefasst. Gemeinsam mit einem festgelegten Salt kann daraus ein 256-Bit AES Schlüssel erzeugt werden, mit dem der VMK entsiegelt wird. Der Wiederherstellungsschlüssel muss bei der Einrichtung vom BitLocker immer erzeugt und zwingend auf ein Medium außerhalb der zu verschlüsselnden BitLocker-Partition gespeichert werden. Dies kann ein Wechseldatenträger oder ein Onlinespeicher sein. Für dieselbe Partition können mehrere Wiederherstellungsschlüssel angelegt werden. [29]

Konnte mit einer der oben genannten Authentifizierungsmaßnahmen erfolgreich der VMK geöffnet werden, so erhält man den entsiegelten VMK.

Volume Master Key (VMK), entsiegelt:

In der **Tabelle 3** ist der Aufbau eines beispielhaften entsiegelten VMK dargestellt. An diesen sollen die einzelnen Bestandteile des Schlüssels erläutert werden, da diese für den späteren Angriff von Bedeutung sein werden.

Offset	Wert															
0	2c	00	00	00	01	00	00	00	03	20	00	00	e4	f8	31	ba
16	6b	26	83	c2	28	25	13	54	f3	06	2b	ad	a9	23	25	77
32	2f	1f	4b	da	84	eb	49	d5	dd	2a	7f	35				

**Tabelle 3:** Aufbau entsiegelter VMK

Der entsiegelte VMK ist immer 44 Bytes lang und verfügt über eine feste 12 Byte lange Signatur (dunkelblau). Die Bytes eins und zwei geben die Länge des

gesamten Schlüssels an. Diese ist aktuell immer 44 Bytes, weshalb die Signatur auch mit dem Hexadezimalwert 0x2c 0x00 beginnt. Anschließend folgt ein weiteres Byte, das die Werte 0x00 bis 0x06 annehmen kann. An der Stelle vier und fünf steht die Versionsnummer des VMK. Im obigen Beispiel besitzt der VMK die Versionsnummer eins. Es folgen zwei weitere Bytes, welche reserviert sind und die Werte 0x00 0x00 besitzen. Die Bytes mit dem Offset acht und neun geben an, welcher Schutz des VMKs gewählt wurde. Der Wert 0x00 0x03 steht für den TPM-Schutz. Die Signatur endet mit drei weiteren reservierten Bytes, welche immer den Wert 0x20, 0x00 und 0x00 besitzen. Direkt im Anschluss der Signatur folgt der 256-Bit große entsiegelte VMK (blau).

Mit dem entsiegelten VMK und weiteren Informationen aus den FVE Meta Datenblöcken kann aus diesem der FVEK extrahiert werden.

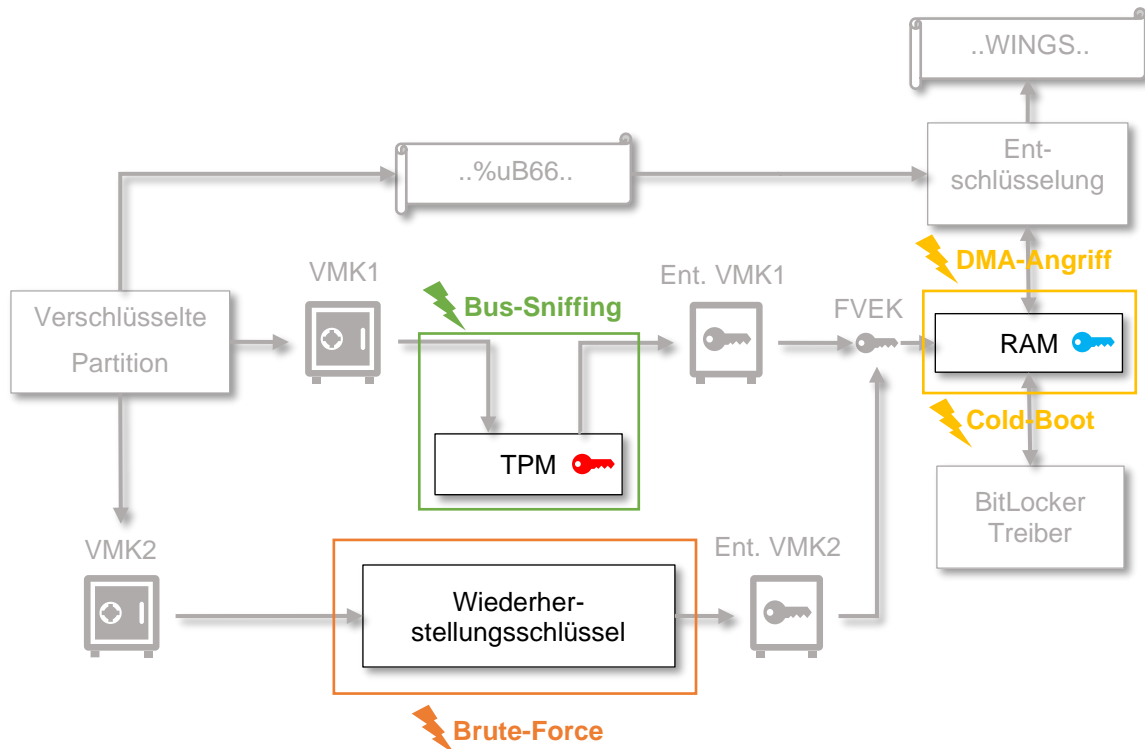
#### Full Volume Encryption Key (FVEK):

Der FVEK dient dazu, die Roh-Daten auf der Festplatte zu ver- bzw. entschlüsseln. Die Ver- und Entschlüsselung der Rohdaten geschieht in der aktuellen BitLocker Version standardmäßig mit dem FVEK über den XTS-AES 128-Bit Algorithmus, wobei der FVEK in diesem Fall eine Länge von 256-Bit besitzt. Es ist allerdings auch möglich, den Verschlüsselungstyp auf AES 256-Bit zu ändern, was Einfluss auf die Länge des FVEK hat. Die Ver- und Entschlüsselung geschieht im laufenden Betrieb in Echtzeit. Um dies gewährleisten zu können, benötigt BitLocker permanenten Zugriff auf den Schlüssel. Dazu muss dieser in den Arbeitsspeicher des Systems geladen werden. [27, S. 18-19]

Auf den FVEK im Arbeitsspeicher hat BitLocker permanenten Zugriff und kann so die zu entschlüsselnden Daten verarbeiten. Für den Benutzer verhält sich das Windows Betriebssystem genauso, als wenn keine Verschlüsselung aktiviert wäre.

## 2.3 Theoretische Angriffsvektoren

Nach dem die Festplattenverschlüsselung BitLocker beschrieben wurde, sollen an dieser Stelle mögliche Angriffsvektoren aufgezeigt werden. Hierzu wurde in der **Abbildung 7** noch einmal der BitLocker Startprozess mit TPM in vereinfachter Form dargestellt.



**Abbildung 7:** BitLocker Angriffsvektoren

Aus dieser Darstellung ergeben sich drei mögliche Angriffspunkte. Der erste Angriff zielt auf den Wiederherstellungsschlüssel ab. Durch Ausprobieren des gesamten Schlüsselraumes kann versucht werden, den VMK2 zu entsiegeln. Der zweite mögliche Angriff beruht darauf, dass der FVEK nach einem erfolgreichen Systemstart in den Arbeitsspeicher des Systems geladen wird. Durch ein erfolgreiches Auslesen des Arbeitsspeichers könnte ein Angreifer in den Besitz des FVEK gelangen. Der dritte hier identifizierte Angriff zielt auf das TPM selbst ab. Während der Kommunikation des Systems mit dem TPM werden Schlüssel ausgetauscht, die möglicherweise mitgelesen werden können. Die daraus

resultierenden Angriffsvektoren Brute-Force, Cold-Boot / DMA-Angriff und TPM Bus-Sniffing, sollen nachfolgend vorgestellt werden.

### **2.3.1 Brute-Force gegen den Wiederherstellungsschlüssel**

Mit jeder BitLocker Verschlüsselung wird immer auch automatisch ein zusätzlicher VMK für die Schutzmaßnahme des Wiederherstellungsschlüssel angelegt. Der Schlüssel verfügt über eine feste Länge, mit vorgeschriebenem Zeichensatz und einem Salt im Klartext auf der verschlüsselten BitLocker-Partition. Die feste Länge und der vorgegebene Zeichensatz beschränken den möglichen Schlüsselraum, während der Salt die Anzahl der Iterationen für die Schlüsselerzeugung vorgibt. Da das Verfahren zur Erzeugung des AES-Schlüssels zum Entsiegeln des VMK bekannt ist, besteht die Möglichkeit, durch Ausprobieren verschiedener Wiederherstellungsschlüssel-Kombinationen den passenden Schlüssel zu finden. Das Ausprobieren aller möglichen Kombination aus dem Schlüsselraum wird als Brute-Force Angriff bezeichnet.

Diese Form des Angriffes, gegen BitLocker, wurde erstmals von E. Agostini und M. Bernaschi in der wissenschaftlichen Veröffentlichung „BitCracker: BitLocker meets GPUs“ [17] vorgestellt. Ihnen ist es gelungen, eine Methode darzustellen, mit der alle notwendigen Informationen für diesen Angriff von der BitLocker verschlüsselten Partition extrahiert werden und aus diesen zusammen mit einem Passwortkandidaten ein AES-Schlüssel erzeugt wird. Mit dem erzeugten AES-Schlüssel wird nun versucht den VMK zu entsiegeln. Dabei können sowohl VMKs mit Passwort als auch mit Wiederherstellungsschlüssel als Schutzmechanismus angegriffen werden. Verfügt der entsiegelte VMK über das richtige Format (Signatur) wird dieser als valide angesehen, und der richtige Schlüssel wurde gefunden. Mit Hilfe von Grafikkartenunterstützung ist es dem Programm BitCracker so möglich, mehrere Passwortkandidaten pro Sekunde zu überprüfen. BitCracker wurde mittlerweile in den Passwort-Cracker „John The Ripper“ mit aufgenommen.

### 2.3.2 FVEK im Arbeitsspeicher

Ein weiterer potenzieller Angriffspunkt in der BitLocker Verschlüsselung stellt der FVEK dar. Dieser wird, nach dem er erfolgreich aus dem VMK gewonnen wurde, in den Arbeitsspeicher geladen. Ist man in der Lage ein Abbild des Arbeitsspeichers zu laden und abzuspeichern, so gelangt man auch in den Besitz des FVEK und umgeht damit sämtliche Schutzmaßnahmen von BitLocker. Das Erstellen des Speicherabbildes kann sowohl softwareseitig als auch mit entsprechender Hardwaremanipulation erfolgen.

Bei einem geöffneten Windows Betriebssystem, bei dem der Benutzer über Admin-Rechte verfügt, kann mit entsprechenden Programmen ein Speicherabbild erstellt werden. Ein gängiges Programm dazu ist z.B. DumpIt. [31, S. 122]

Ist das System gesperrt oder verfügt der Benutzer über keine Adminrechte, besteht weiterhin die Möglichkeit eines Hardware-Angriffes.

#### 2.3.2.1 DMA Angriff

Über ein Bus-System werden bei laufendem Betrieb in kurzer Zeit sehr große Datenmengen zwischen E/A-Geräten (z.B. PCIe Geräten) und dem Arbeitsspeicher ausgetauscht. Dies geschah früher nicht direkt, sondern wurde von dem Prozessor gesteuert. Dieser musste zunächst die Daten der E/A-Geräte in ein Register laden, verarbeiten und anschließend weiter verteilen. Diese Art des Speicherzugriffs ist ineffizient, da immer erst auf die Verarbeitung des Prozessors gewartet werden muss. Aus diesem Grund wurde das „Bus Mastering“ eingeführt, mit dem es möglich wurde, dass Geräte die Kontrolle über den Bus selbst übernehmen und direkt mit dem Arbeitsspeicher kommunizieren können. Der direkte Zugriff auf den Speicher erhöhte zwar den Datenfluss, ermöglichte aber auch neue Angriffe. Angreifern ist so möglich geworden, durch böartige Hardware, direkt Zugriff auf den gesamten Arbeitsspeicher zu erlangen und diesen zu lesen oder zu verändern. [32] Solche Angriffe werden als Direct Memory Access (DMA) Attack (dt. DMA-Angriff) bezeichnet. Ein Beispiel für solch

einen Angriff ist das PCILeech Projekt von Ulf Frisk. [33] Bei diesem kann über verschiedene Schnittstellen, wie z.B. PCIe und Thunderbolt, versucht werden, den Inhalt des Arbeitsspeichers vom Zielrechner auf den angreifenden Rechner zu übertragen.

### 2.3.2.2 Cold Boot Angriff

Eine weitere Möglichkeit an die Daten im Arbeitsspeicher zu gelangen, stellt der Cold Boot Angriff dar. Dieser Angriff wurde erstmals 2008 von Princeton University in der wissenschaftlichen Veröffentlichung „Lest We Remember: Cold Boot Attacks on Encryption Keys“ [34] vorgestellt.



**Abbildung 8:** Cold Boot - eingefrorener DDR2 Speicher

Bei den meisten der heute erhältlichen Computern wird der sogenannte Dynamic Random Access Memory (DRAM) verbaut. Die Technik des Cold Boot Angriffs basiert auf den physikalischen Eigenschaften dieses Speichers, genauer auf dessen Speicherzellen. Eine Speicherzelle im DRAM wird durch einen Transistor und einen Kondensator repräsentiert. Die Ladung des Kondensators bestimmt den Zustand der Speicherzelle. Dabei nimmt die Speicherzelle geladen beispielsweise den Zustand „1“ und ungeladen den Zustand „0“ an. Aufgrund von Leckströmen und der geringen Kapazität des Kondensators verliert die Speicherzelle schnell ihren Zustand. Um das zu unterbinden, muss die

Speicherzelle in regelmäßigen Abständen aktualisiert werden, in dem eine entsprechende Spannung angelegt wird.

Wird der Computer ausgeschaltet, ist er stromlos und die Speicherzellen des DRAM verlieren in wenigen Sekunden ihre Zustände. Je länger die Speicherzellen nicht mehr versorgt werden, desto größer ist der Datenverfall.

Bei dem Cold Boot Angriff wird versucht, das System mit einem vorbereiteten minimalen Betriebssystem neu zu starten, um über dieses an ein Speicherabbild zu gelangen. Durch das Laden eines alternativen Betriebssystems, auf welchem alle benötigten Rechte und Programme zum Erstellen eines Speicherabbildes vorhanden sind, wird zwangsläufig ein Teil des Arbeitsspeichers überschrieben. Es ist demnach wichtig, dass das alternative Betriebssystem so klein wie möglich ist, um möglichst viele Daten vom letzten Systemstart zu erhalten. Wird ein Fremdbooten jedoch durch das System unterbunden, so kann in einem weiteren Schritt versucht werden, den Arbeitsspeicher vom Zielrechner zu lösen und diesen auf ein vorbereitetes kompromittiertes System umzusetzen. Von diesem System kann dann wieder das vorbereitete minimale Betriebssystem gestartet werden, um an das Speicherabbild zu gelangen. Das Problem bei diesem Angriff stellt die Zeitspanne dar, in der das System stromlos ist und der Datenverfall einsetzt. Um den Datenverfall so gering wie möglich zu halten, müssen die Leckströme verlangsamt werden. Dies kann erreicht werden, indem die Teilchenbewegung im Speichermodul reduziert wird. Die Teilchenbewegung hat eine direkte Abhängigkeit zur Temperatur und setzt theoretisch bei  $-273,15^{\circ}\text{C}$  aus. Der Cold Boot Angriff nutzt dies und versucht, die Speicherzelle herunterzukühlen, um die Leckströme zu reduzieren. Erreicht wird dies in der Regel mit entsprechenden Kältesprays. Hiermit werden allerdings nur Temperaturen bis zu  $-50^{\circ}\text{C}$  erreicht, weshalb Leckströme nur reduziert, aber nie ausgeschlossen werden können. [34] [35]

### **2.3.3 Mitlesen am TPM**

Der dritte theoretisch möglich Angriff ist das Mitlesen der Kommunikation am TPM, das sogenannte TPM Bus-Sniffing. Zum Entsiegeln muss der versiegelte



VMK zunächst an das TPM gesendet werden. Dort wird der VMK mit Hilfe des SRK entsiegelt und kann dort von der CPU wieder abgerufen werden. Es entsteht somit eine Kommunikation zwischen System und dem TPM. Diese Kommunikation verläuft, wie in 2.1.1 bereits erwähnt, über ein Bus System. 2019 veröffentlichte D. Andzakovic einen Beitrag, in dem er zeigte, dass diese Kommunikation zwischen dem System und dem TPM unverschlüsselt stattfindet. [36] Dies eröffnet die Möglichkeit, den entsiegelten VMK mitzulesen, wenn das System diesen vom TPM abrufen.

Um zu verstehen wie eine Kommunikation über einen Bus abläuft, sollen der LPC und eSPI-Bus nachfolgend vorgestellt werden.

### 2.3.3.1 Low Pin Count (LPC)

Der Low Pin Count (LPC) Bus wurde 1998 von Intel entwickelt und es handelt sich hierbei um einen Bus, welcher mit nur wenigen Anschlüssen auskommt. Ziel dieser Entwicklung war es, die bisherige Industry Standard Architecture (ISA) zu ersetzen. Dabei sollten alle Funktionalitäten des Vorgängers beibehalten, mit neuen Funktionen erweitert und gleichzeitig die Anzahl der benötigten Signalleitungen reduziert werden. Zu den geerbten Funktionen von ISA gehören:

- Dedizierter E/A Adressraum
- Memory Mapped I/O (MMIO)
- Direct Memory Access (DMA)

Mit dem LPC-Standard wurde die Unterstützung eines einfachen „Bus-Mastering“ Protokolls neu hinzugefügt. Dieses soll den bisherigen direkten Speicherzugriff (DMA) im Stil von ISA, in bestimmten Situationen, ersetzen. [37, S. 14]

Der LPC-Bus dient ausschließlich der Kommunikation zwischen Komponenten auf dem Motherboard. Für diesen Standard sind keine Anschlussmöglichkeiten in Form von Steckplätzen, wie beispielsweise ISA sie noch hatte, vorgesehen. Mit Einzug des Peripheral Component Interconnect (PCI) Bus waren diese auch nicht mehr nötig. Gesteuert wird der LPC-Bus von der Southbridge bzw. dem Plattform Controller Hub (PCH), die als Host des Bus-Systems agieren. Der Host

ist direkt oder über einen vorgeschalteten Controller mit der CPU verbunden. Geräte, die über physische Leitungen mit dem LPC-Bus verbunden sind, werden als Peripheriegeräte bezeichnet. In der Regel benötigen diese Peripheriegeräte für die Datenübertragung nur eine geringe Bandbreite oder sind nicht latenzkritisch. Typische LPC-Bus Peripheriegeräte sind die Firmware-ROM, Legacy Geräte wie PS/2 Tastatur und Maus, der Floppy Disk Controller und das TPM. [37, S. 11]

In der einfachsten Konfiguration benötigt der LPC-Bus nur sieben (sechs Signalleitungen + Bus-Takt) und in der erweiterten Konfiguration 13 Anschlüsse. Im Vergleich dazu benötigte ein voll ausgebauter 8-Bit ISA-Bus 54 Datenleitungen und ein 16-Bit ISA-Bus sogar 88 Signalleitungen. Bei gleicher Funktionalität konnte der LPC-Bus viele Signalleitungen einsparen. Dies führte zu reduzierten Kosten, kleineren E/A Schaltkreisen und mehr Platz auf der Hauptplatine, auf die der LPC-Bus verwendet wird. Erreicht wurde die Reduzierung durch eine einfache bidirektionale Architektur mit Zeitmultiplexing auf den Datenleitungen. Zusätzlich wurde der Takt vom LPC-Bus von ehemals 8,33 MHz (ISA) auf bis zu 33,3 MHz angehoben. Der Pegel des LPC-Bus entspricht in der Regel 3,3V bzw. 5V. [37, S. 49-50]

Damit entspricht der LPC-Bus weitestgehend den technischen Spezifikationen und Anforderungen des PCI-Bus und ist über den Host, genauer mit der PCI-to-LPC Bridge, mit diesem verbunden.

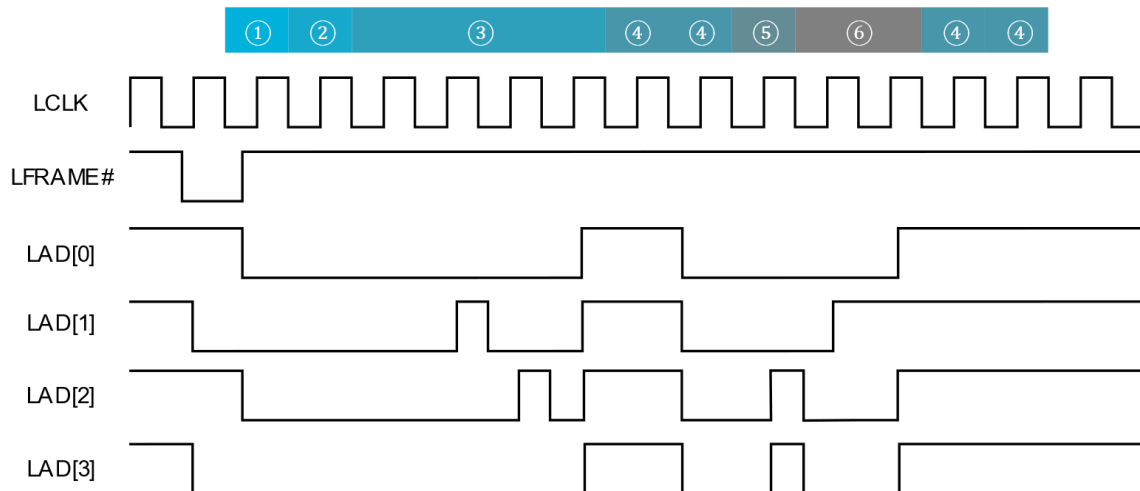
Wie oben erwähnt, besteht der LPC-Bus in der einfachsten Konfiguration aus sieben Signalleitungen. Diese sind aus Sicht des Host in der **Tabelle 4** dargestellt.

Signalleitung	Richtung	Beschreibung
LCLK	Ausgang	Bus-Takt
LFRAME#	Ausgang	Beginn eines Bus-Zyklus
LRESET#	Ausgang	Rücksetzsignal
LAD[0]	Bidirektional	Übertragung der Daten
LAD[1]	Bidirektional	Übertragung der Daten
LAD[2]	Bidirektional	Übertragung der Daten
LAD[3]	Bidirektional	Übertragung der Daten

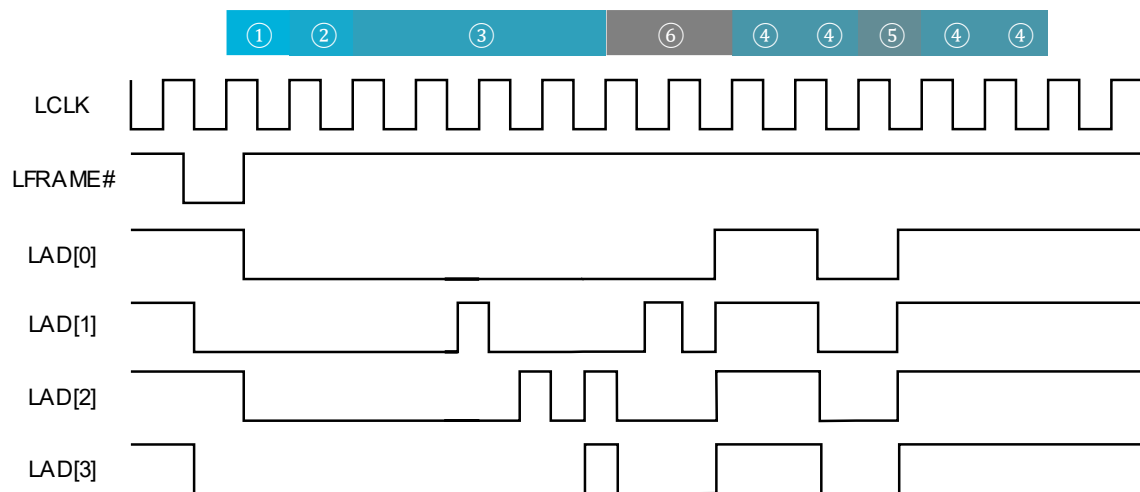
**Tabelle 4:** Signalleitungen LPC, einfache Ausführung [37, S. 9]

Die Signale der Ausgangsleitungen werden vom Host erzeugt und von der Peripherie nur gelesen. Die Datenleitungen LAD[0] bis LAD[3], sowie die LFRAME# Leitung des LPC-Bus verfügen über 15-100 k $\Omega$  Pull-Up-Widerstände, was dazu führt, dass diese als logisches Signal immer „high“ anliegen haben, wenn sie nicht aktiv von einem Gerät angesteuert werden. Sie beziehen sich, mit Ausnahme von LRESET#, immer auf die High-Low-Flanke des Bus-Taktes.

Wie oben erwähnt, verfügt der LPC-Bus in der erweiterten Konfiguration über sechs zusätzliche Signalleitungen. Diese dienen u.a. für Interrupts und DMA-Anfragen. Es handelt sich dabei um die Signalleitungen LDRQ#, SERIRQ#, CLKRUN#, LPME#, LPCPD# und LSMI#. Da diese Signalleitungen für den Bus-Sniffing Angriff nicht benötigt werden, sind diese hier nur vollständigheitshalber erwähnt. In der **Abbildung 9** und **Abbildung 10** ist beispielhaft der Verlauf von zwei LPC-Frames dargestellt. Sie stellen jeweils einen E/A Lese- und einen E/A Schreib-Zyklus dar. Anhand des Lese-Zyklus soll nachfolgend die einzelnen Phasen, die ein LPC-Frame durchläuft, vorgestellt werden.



**Abbildung 9:** LPC-Frame – E/A Lese-Zyklus



**Abbildung 10:** LPC-Frame, E/A Schreib-Zyklus

### 1. START-Phase

Ein LPC-Frame beginnt mit der START-Phase. Eingeleitet wird die START-Phase dadurch, dass das LFRAME# Signal für einen Takt (Normal Timing) oder für mehrere aufeinanderfolgende Takte (Extended Timing) auf „low“ gesetzt wird. Während dieser Phase zeigen die Datenleitungen den Transaktionstyp an. In der

**Tabelle 5** sind alle möglichen Transaktionstypen aufgeführt. Einige davon sind für entsprechende Anwendungen reserviert.

LAD[3]	LAD[2]	LAD[1]	LAD[0]	Hex	Beschreibung
0	0	0	0	0x0	Start eines Bus-Zyklus
0	0	0	1	0x1	Reserviert
0	0	1	0	0x2	Rückmeldung für Bus Master 0
0	0	1	1	0x3	Rückmeldung für Bus Master 1
0	1	0	0	0x4	Reserviert
0	1	0	1	0x5	Start eines TPM-Zyklus
0	1	1	0	0x6	Reserviert
0	1	1	1	0x7	Reserviert
1	0	0	0	0x8	Reserviert
1	0	0	1	0x9	Reserviert
1	0	1	0	0xA	Reserviert
1	0	1	1	0xB	Reserviert
1	1	0	0	0xC	Reserviert
1	1	0	1	0xD	Start zum Lesen des Firmware Memory
1	1	1	0	0xE	Start zum Schreiben der Firmware Memory
1	1	1	1	0xF	Stopp bzw. Abbruch des Bus-Zyklus

**Tabelle 5:** LPC-Zyklus, START-Phase [37, S. 15]

## 2. CYCLE TYPE/DIRECTION-Phase

Mit dem nachfolgenden Takt steuert das LFRAME# Signal zurück auf „high“ und leitet damit die CYCLE TYPE/DIRECTION-Phase ein. Diese Phase dauert einen

Takt-Zyklus, signalisiert den Typ der Übertragung (E/A, MMIO, DMA) und die Richtung (lesen, schreiben) des LPC-Frames. Die Signalleitungen können nachfolgende Werte annehmen, wobei LAD[0] vom Host, in dieser Phase, immer auf den Wert „low“ gehalten wird.

LAD[3]	LAD[2]	LAD[1]	LAD[0]	Hex	Beschreibung
0	0	0	0	0x0	E/A lesen
0	0	1	0	0x2	E/A schreiben
0	1	0	0	0x4	MMIO lesen
0	1	1	0	0x6	MMIO schreiben
1	0	0	0	0x8	DMA lesen
1	0	1	0	0xA	DMA schreiben
1	1	0	0	0xC	Reserviert
1	1	1	0	0xE	Reserviert

**Tabelle 6:** LPC-Zyklus, Cycle Type/Direction-Phase [37, S. 15]

### 3. ADDRESS-Phase

In dieser Phase wird das Ziel des LPC-Frames adressiert. Sie dauert 4-Takte für 16-Bit Adressen und 8-Takte für 32-Bit Adressen. Die Länge der Adresse ist abhängig vom Typ der Übertragung. E/A Typen verlangen 16-Bit Adressen, während MMIO Typen eine 32-Bit-Adressen benötigen. Beim Lesen der Adresse durch einen Dekodierer muss beachtet werden, dass die Übertragung mit dem höchstwertigen Nibble beginnt.

### 4. TAR-Phase

Es folgen zwei Turnarounds (TAR). Diese Phase signalisiert, dass der Besitz der Datenleitungen an das entsprechende LPC-Gerät übertragen bzw. zurückgegeben wird. Ein TAR dauert einen Takt lang und wird angezeigt, indem alle Datenleitungen auf „high“ gesetzt werden.

## 5. SYNC-Phase

Es liegt nun in der Verantwortung des Ziel-Geräts, den LPC-Buszyklus während der SYNC-Phase zu bestätigen oder mit einem Fehler abubrechen. Die SYNC-Phase dauert einen Takt lang. Bei Bedarf kann ein Peripheriegerät zusätzliche Wartezustände einfügen, indem es die Datenleitungen auf spezielle "Warten kurz"- oder "Warten lang"-Werte treibt, bis die Bustransaktion fortgesetzt werden kann. Erhält der Host drei Mal in Folge auf allen Datenleitungen einen „high“ Wert zurück, geht er davon aus, dass kein Gerät antwortet und bricht die Übertragung ab. In der **Tabelle 7** sind alle möglichen Antworten des Ziel-Gerätes aufgeführt.

LAD[3]	LAD[2]	LAD[1]	LAD[0]	Hex	Beschreibung
0	0	0	0	0x0	Transaktion erfolgreich
0	0	0	1	0x1	Reserviert
0	0	1	0	0x2	Reserviert
0	0	1	1	0x3	Reserviert
0	1	0	0	0x4	Reserviert
0	1	0	1	0x5	Kurzes Warten
0	1	1	0	0x6	Langes Warten
0	1	1	1	0x7	Reserviert
1	0	0	0	0x8	Reserviert
1	0	0	1	0x9	Transaktion erfolgreich (DMA)
1	0	1	0	0xA	Transaktion fehlerhaft
1	0	1	1	0xB	Reserviert
1	1	0	0	0xC	Reserviert
1	1	0	1	0xD	Reserviert
1	1	1	0	0xE	Reserviert
1	1	1	1	0xF	Reserviert

**Tabelle 7:** LPC-Zyklus, SYNC [37, S. 17]

## 6. DATA-Phase

Mit den folgenden zwei Takten erfolgt die Datenübertragung vom Ziel-Gerät zum Host. Mit einem LPC-Frame kann jeweils ein Byte (8-Bit) übertragen werden. Sind die zu lesenden Daten länger, müssen diese auf mehrere LPC-Frames aufgeteilt werden. Auf dem ersten Takt wird das niederwertige Nibble (Bit 0-3) und auf dem zweiten Takt das höherwertige Nibble (Bit 4-7) übertragen. Mit einer weiteren folgenden TAR Phase wird der Besitz über die Datenleitungen wieder an den Host übertragen und der LPC Lese-Zyklus beendet. Bei dem in **Abbildung 9** abgebildeten LPC-Frame handelt es sich nach Vorgabe der START-Phase um einen TPM-Bus-Zyklus, der eine E/A Lese-Transaktion ausführt. Bei der Transaktion wird von der Zieladresse 0000 0000 0010 0100 (0x0024) gelesen. Nach der TAR-Phase antwortet das Zielgerät erfolgreich zurück und beginnt mit der Übertragung des angeforderten Byte. Dieses besitzt den Wert 0010 0011 (0x2C).

Im Gegensatz zum LPC-Lese-Zyklus folgt bei einem Schreib-Zyklus nach der ADDRESS-Phase die DATA-Phase. Dann folgt die TAR-Phase, die Daten werden geschrieben, der Erfolg der Transaktion zurückgemeldet und mit einer weiteren TAR-Phase der LPC-Schreib-Zyklus beendet. Bis zur Version 1.1 von TPM existierte keine einheitliche Registerschnittstelle zwischen dem TPM und der Anwendung, die darauf zurückgreift. Dies hatte zur Folge, dass es je nach Hersteller und Entwickler unterschiedliche Schnittstellen gab, die untereinander mehr oder weniger kompatibel waren. Erst mit der Einführung von TPM 1.2 wurde von der TCG der einheitliche TPM Interface Standard (TIS) geschaffen. Dieser definiert u.a. die Schnittstelle zwischen Anwendung und TPM. Darüber hinaus legt er auch den Ablauf eines LPC-Bus-Zyklus fest, der für die Kommunikation notwendig ist. [9, S. 182] Allgemein gleicht der LPC-Bus-Zyklus für die TPM Kommunikation dem des in **Abbildung 9** dargestellten E/A Lese-Zyklus. Die START-Phase beginnt mit dem Wert 0x5 und leitet damit einen TPM-Zyklus ein. Die Adressen der LPC-Frames, die auf das Register mit dem möglichen entsiegelten VMK im TPM zeigen, besitzen die Werte 0x0024, 0x0025, 0x0026 oder 0x0027. [5, S. 46, 67] Welche von den vier Adressen verwendet wird, ist hersteller- und chipabhängig.



2016 wurde der LPC-Bus durch den Intel Enhanced Serial Peripheral Interface (eSPI) Bus abgelöst.

### 2.3.3.2 *Enhanced Serial Peripheral Interface (eSPI)*

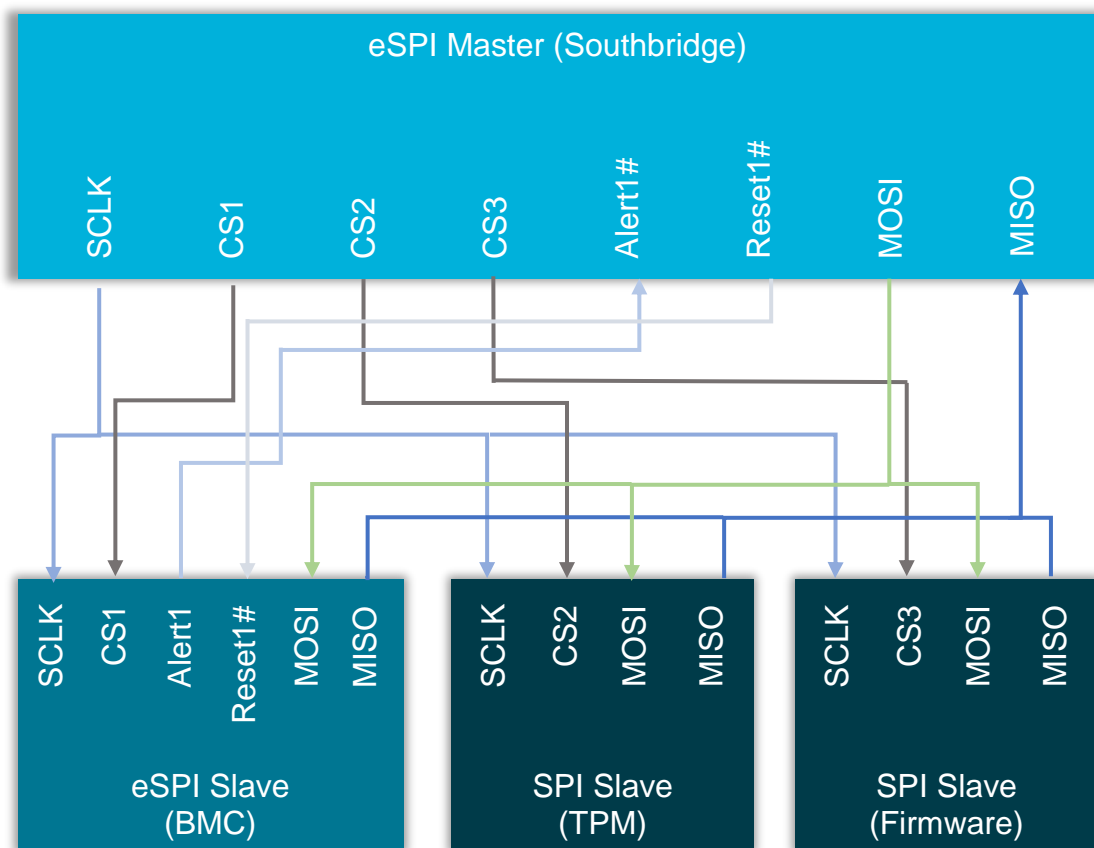
Der SPI-Bus, auf dem der eSPI-Bus basiert, wurde schon 1987 von Motorola entwickelt. [38] Mit der Einführung von eSPI sollen Schwächen des mittlerweile in die Jahre gekommenen LPC-Busses beseitigt werden. Zu den größten Schwächen zählt die für heutige Verhältnisse hohe Anzahl an Datenleitungen (7 bzw. 13) und die festgelegte Übertragungsrate von 132 Mbps bei einer Taktfrequenz von 33 MHz. Diese wirkt sich vor allem bei neuen Schnittstellen Standards wie PCI Express und USB3 limitierend aus. Dem Namen des Busses entsprechend, handelt es sich bei eSPI um eine Erweiterung des SPI-Bus. Dabei orientiert eSPI sich an den technischen Spezifikationen wie z.B. dem Takt von SPI, unterscheidet sich aber beim Übertragungsprotokoll von SPI. [39, S. 9]

Als Nachfolger von LPC sollte eSPI dessen Funktionalitäten beibehalten. Lediglich 8237 DMA und Firmware Hub (FWH) werden von eSPI nicht mehr unterstützt. Zu den weiteren Anforderungen an eSPI gehören ein niedriger anpassbarer Energieverbrauch des Busses in jedem System Status (S0-S5), eine Reduzierung der Datenleitungen, eine Erhöhung der Bus Bandbreite, eine der Anwendung entsprechende anpassbare Bus Bandbreite, ein auf Memory Mapping basierendes Flash-Sharing sowie das Verwenden eines E/A Buffers mit geringer Spannung (1,8V). [39, S. 12-13]

Das Master-Slave Prinzip stellt die Grundlage für die Kommunikation bei eSPI dar. Der eSPI Master steuert den Befehls- und Datenfluss zwischen sich und einem oder mehreren eSPI Slaves. Dabei kann es immer nur einen eSPI Master im Bus Netz geben. Entsprechend der Anzahl der eSPI-Slaves im Bus Netz erfolgt eine Unterteilung in Master-Single-Slave oder Master-Multiple-Slaves Betrieb. Als eSPI Master in einem Computersystem fungiert in der Regel der Southbridge Controller. Typische eSPI Slaves sind z.B. der Embedded Controller (EC), der Baseboard Management Controller (BMC) sowie Super I/O Controller (SIO). Eine Besonderheit von eSPI ist es, dass weiterhin klassische SPI-Slaves

an einen eSPI-Bus verwendet werden können. Ein SPI-Slave an einem eSPI-Bus ist beispielsweise das TPM. Dieser teilt sich denselben Takt und dieselben Datenleitungen mit den anderen eSPI-Slaves. Dabei kommuniziert der Nicht-eSPI Slave allerdings weiter über klassische SPI-Protokolle mit dem eSPI Master. [39, S. 14-21]

In der **Abbildung 11** ist beispielhaft ein eSPI-Bus mit einem Master und mehreren (e)SPI-Slaves dargestellt.



**Abbildung 11:** eSPI-Bus mit Standard E/A

Der eSPI-Bus besteht aus sechs Signalleitungen, wobei für den Datenaustausch lediglich vier Signalleitungen benötigt werden. Im Vergleich zum LPC-Bus können so zwei Signalleitungen eingespart werden, was kleinere Komponenten und mehr Platz auf dem Mainboard ermöglicht. Die verwendeten Signalleitungen eines eSPI-Busses aus Sicht des eSPI-Master sind in der **Tabelle 8** dargestellt.

Signalleitung	Richtung	Beschreibung
SCKL	Ausgang	Bus Takt, wird vom eSPI-Master erzeugt und vom Slave gelesen
E/A [n:0] (MOSI/MISO)	Bidirektional	Diese Signalleitungen dienen der Übertragung der Daten zwischen eSPI Master und (e)SPI Slave. Die Anzahl der Datenleitungen wird durch n repräsentiert und kann die Werte 1 oder 3 annehmen. Es wird unterschieden in Standard E/A (n=0), Dual E/A (n=1) und Quad E/A (n=3) Übertragung.
CS#	Ausgang	Chip Select#, wenn CS# auf „low“ getrieben wird, ist die Kommunikation mit diesem Slave aktiviert. Dies verhindert, dass mehrere Slaves den Bus gleichzeitig benutzen. Jeder Slave verfügt über eine eigene CS# Leitung.
Alert#	Eingang	Alert# wird vom Slave verwendet, um den Alarm Dienst beim Master auszulösen.
Reset#	Ein- oder Ausgang	Setzt die eSPI-Schnittstelle für den Master und den Slave zurück.

**Table 8:** eSPI-Signalleitungen

Im Vergleich zu SPI wurden bei eSPI die Signalleitungen Alert# und Reset# neu hinzugefügt, sowie die Möglichkeit von zwei zusätzlichen bidirektionalen Datenleitungen, die drei Übertragungsarten ermöglichen. SPI verfügt lediglich über den Standard E/A Übertragungs-Modus, der aber auch bei eSPI weiter angewandt werden kann. In diesem Modus existieren zwischen den Teilnehmern zwei Datenleitungen, die unidirektional genutzt werden. Die Datenleitung vom eSPI-Master zum (e)SPI-Slave wird als Master-Output-Slave-Input (MOSI)

bezeichnet, während die Datenleitung vom (e)SPI-Slave zum eSPI-Master als Master-Input-Slave-Output (MISO) bezeichnet wird. In den anderen eSPI Übertragungsarten werden alle Datenleitungen bidirektional betrieben, was eine höhere Datenübertragung auf bis zu vier Datenleitungen ermöglicht.

Da es sich beim TPM um einen klassischen SPI-Slave an einem eSPI-Bus handelt und dieser über das klassische SPI-Protokoll kommuniziert, wird in den folgenden Abschnitten die Kommunikation von SPI näher beschrieben.

SPI ist ein frei konfigurierbarer Bus, dessen Einstellungen in drei Registern erfolgen. Jede SPI-Komponente besteht dabei aus einem Statusregister (SPSR), einem Datenregister (SPDR) und einem Konfigurationsregister (SPCR). Das Statusregister ermöglicht das Setzen von Interrupts und Kollisions-Flags, sowie eine Erhöhung der Taktfrequenz. Das Datenregister dient dem Datenaustausch zwischen Master und Slave. Beim Konfigurationsregister werden die Einstellungen für die Konfiguration der Komponente im SPI-Bus getroffen. Diese müssen für alle Geräte im Bus aufeinander abgestimmt sein. Um den SPI-Bus und seine Arbeitsweise besser zu verstehen, wird zunächst das Konfigurationsregister vorgestellt. Es werden die Einträge des Registers beschrieben und auf die Vorgaben der TCG für TPM mit SPI-Bus eingegangen. Das Konfigurationsregister ist acht Bit groß, wobei jedes Bit einen Eintrag im Register repräsentiert. Diese sind in der **Tabelle 9** vorgestellt. [40, S. 60]





Die Bits 0 und 1 im Konfigurationsregister steuern die Frequenz des Bus Taktes (SCLK). Diese Bits haben nur einen Einfluss, wenn die Komponente als SPI Master betrieben wird. Der SPI-Slave liest lediglich den Takt des SPI-Masters und kann deshalb auf das Setzen dieser beiden Bits verzichten. Die Taktfrequenz ist nach oben immer von der Taktrate des Controllers des SPI-Masters limitiert. Die Taktrate muss so gewählt werden, dass alle angeschlossenen SPI-Slaves in der Lage sind, diese zu verarbeiten. Findet keine Datenübertragung zwischen Master und Slave statt, beträgt die Taktfrequenz null und das Signal wird auf einem gleichbleibenden Pegel gehalten. [41, S. 224]

Bit	Name	Beschreibung
7	SPIE	SPI-Interrupt-Enable Flag, legt fest ob Interrupts erlaubt sind
6	SPE	SPI-Enable, aktiviert bzw. deaktiviert SPI
5	DORD	Data Order, gibt vor, ob die Übertragung mit dem Least Significant Bit (lsb) oder dem Most Significant Bit (msb) beginnt (0=msb, 1=lsb)
4	MSTR	Master/Slave Select, entscheidet ob Gerät als Master oder Slave konfiguriert ist (0=Slave, 1=Master)
3	CPOL	Idle Polarity, setzt die Polarität des Taktsignals im Ruhezustand
2	CPHA	Clock Phase Bit, entscheidet wann Daten in Abgängigkeit des Taktsignals gelesen bzw. gesetzt werden.
1	SPR1	Setzt zusammen mit SPR0 die Taktfrequenz
0	SPR0	Setzt zusammen mit SPR1 die Taktfrequenz

**Tabelle 9:** SPI Konfigurationsregister (SPCR)

Die Register-Bits 2 (CPHA) und 3 (CPOL) entscheiden zusammen über den SPI-Betriebsmodus. SPI verfügt über vier verschiedene Betriebsmodi, welche die Synchronisation des Protokolls festlegen. Der Modus wird gebildet aus der Clock Polarity (CPOL) und der Clock Signal Phase (CPHA). Die CPOL legt den Zustand des Taktes im Leerlauf fest. Hat CPOL den Wert „null“, dann liegt das Taktsignal im Ruhemodus auf dem Pegel „low“. Wenn CPOL den Wert „eins“ zugeordnet bekommt, dann ist das Taktsignal im Ruhemodus auf dem Pegel „high“. CPHA legt fest, wann mit dem Lesen und Setzen der Daten auf dem Taktsignal begonnen werden soll. Dieser Wert muss immer zusammen mit CPOL betrachtet werden. Hat CPHA den Wert „null“, wird mit dem Beginn des Taktsignals gelesen. Das bedeutet bei einem Wert von CPOL „null“ und CPHA „null“, das auf der steigenden Flanke des Taktsignals gelesen und auf der fallenden Flanke die Daten gesetzt werden. Wenn CPOL „eins“ ist, dann wird auf der ersten fallenden

Flanke des Taktes gelesen und auf der steigenden das Bit gesetzt. Umgekehrt verhält es sich dementsprechend bei einem CHPA Wert von „eins“. Welche Werte CPOL und CPHA annehmen können, wird von der verwendeten SPI Komponente bestimmt. In der **Tabelle 10** sind die verschiedenen Betriebsmodi von SPI dargestellt. [41, S. 221-223]

SPI Modus	CPOL im Ruhemodus	CPHA		Beschreibung
0	0	0		Abtasten auf steigender Flanke, setzen auf fallender Flanke
1	0	1		Abtasten auf fallender Flanke, setzen auf steigender Flanke
2	1	0		Abtasten auf steigender Flanke, setzen auf fallender Flanke
3	1	1		Abtasten auf fallender Flanke, setzen auf steigender Flanke

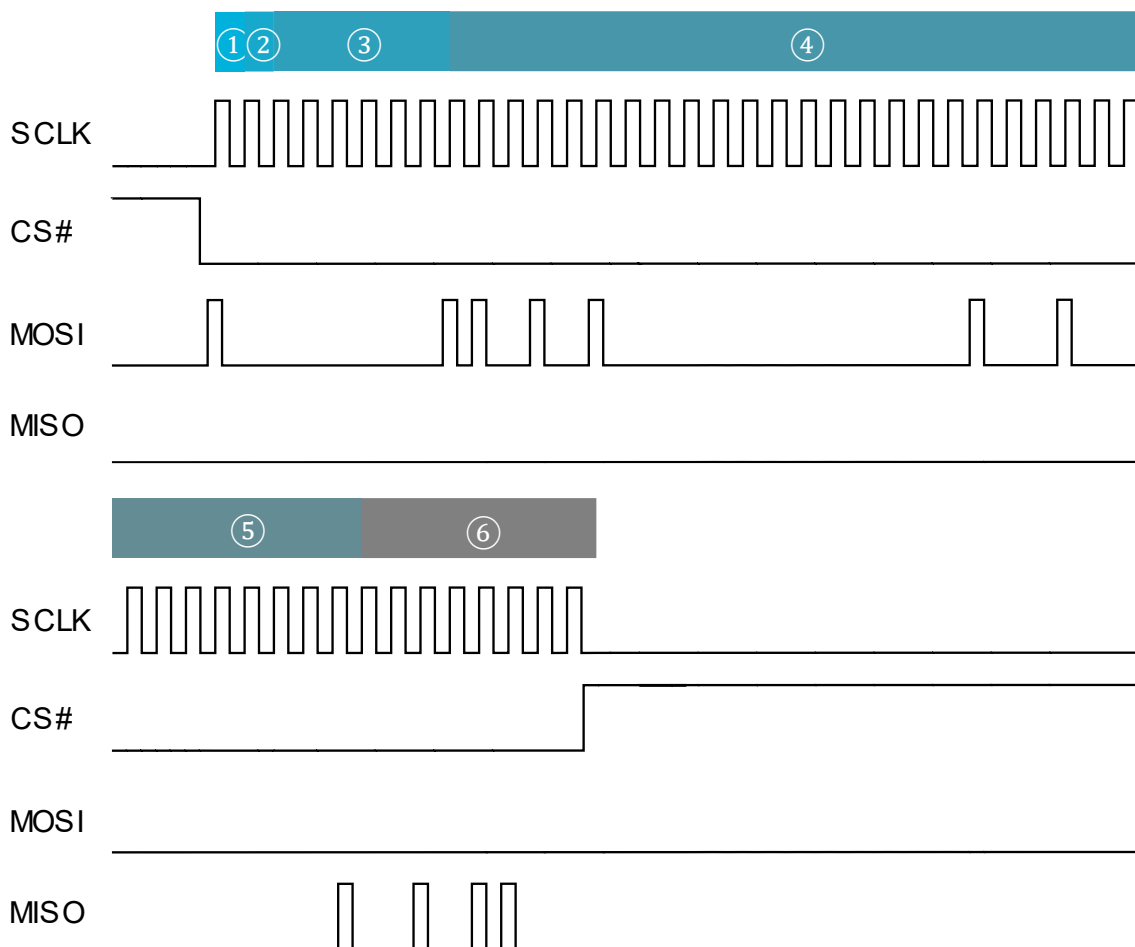
**Tabelle 10:** SPI Modus

Das Bit vier (MSTR) im SCPR Register legt fest, ob es sich bei dem Teilnehmer im SPI-Bus um einen SPI-Master oder um einem SPI-Slave handelt. Wie bereits oben erwähnt, kann es in einem SPI-Bus immer nur einen SPI-Master aber mehrere SPI Slaves geben. Die Synchronisation zwischen Master und Slave erfolgt über eine gemeinsame Taktleitung. Der Takt wird vom Master erzeugt und vom Slave gelesen. Die Entscheidung, welcher Slave über den Bus kommunizieren darf, erfolgt vom Master über die Chip-Select (CS#) Signalleitung. Jeder Slave verfügt über eine eigene CS# Signalleitung, was dazu führt, dass an einem SPI-Master immer nur so viele Slaves verbunden werden können, wie CS#-Signalleitungen am Master vorhanden sind. Zum Ansprechen des jeweiligen Slaves wird dessen CS#-Signalleitung auf den Pegel „low“ gesteuert. [42]

Die Konfiguration der SPI-Schnittstelle sieht weiterhin vor, dass die Reihenfolge, in der die einzelnen Bits gesendet werden, eingestellt werden kann. Dies erfolgt mit dem Bit fünf (DORD) im Konfigurationsregister. Es ist möglich, sowohl mit dem Most Significant Bit (msb) als auch mit dem Least Significant Bit (lsb) die Übertragung zu beginnen. [41, S. 221]

Die letzten beiden Bits im SCPR aktivieren den SPI-Bus (Bit sechs) und legen fest, ob Interrupts während der Übertragung erlaubt sind.

In der **Abbildung 12** ist beispielhaft die Übertragung zwischen einem SPI-Master und einem TPM über einen SPI-Bus dargestellt. Für die Übertragung verwendet das TPM das SPI-Bit-Protokoll, welches nachfolgend anhand der Abbildung erläutert wird.



**Abbildung 12:** SPI-Bit-Protokoll

Das SPI Bit Protokoll kann in sechs Phasen eingeteilt werden, wobei die Phasen eins bis vier vom SPI-Master bedient werden und die Phasen fünf und sechs vom SPI-Slave.

#### 1. TYPE-Phase

In dieser Phase der Kommunikation wurde die CS# Signalleitung am SPI-Slave auf den „low“ Pegel gesetzt und eine Taktfrequenz erzeugt. Mit dem ersten Bit der Übertragung legt der SPI Master den Typ der Transaktion fest. Es kann sich dabei um eine Lese- (0) oder eine Schreib-Transaktion (1) handeln.

#### 2. RESERVED-Phase

Diese Phase dient als Platzhalter. Das Bit, das in dieser Phase erzeugt wird, ist reserviert und kann nicht anders belegt werden.

#### 3. TRANSFER-SIZE-Phase

In der Phase 3 werden sechs Bits übertragen, welche festlegen, wie viele Bytes vom SPI-Slave gesendet bzw. geschrieben werden sollen. Eine Transaktion bedient immer mindestens ein Byte. Die Bitfolge 000000 steht für die Übertragung eines Bytes. Die maximal zulässige Anzahl an Bytes in einer Transaktion beträgt demnach 64 (111111).

#### 4. ADDRESS-Phase

In dieser Phase wird die Zieladresse der Transaktion festgelegt. Die Zieladresse ist 24-Bit lang. Die Übertragung der Adresse erfolgt mit dem höchstwertigen Bit.

#### 5. FLOW-CONTROL-Phase

Da im Vergleich zum LPC-Bus mehr als ein Byte pro Transaktion übertragen werden kann, wird eine Übertragungskontrolle benötigt. SPI bietet selbst keine Kontrollmöglichkeiten, weshalb innerhalb des SPI-Bit-Protokolls die Phase fünf eingeführt wird. Diese Phase erlaubt es dem SPI-Slave Warte-Zustände einzufügen, bevor der SPI Master mit dem Senden bzw. Lesen der Daten fortfährt. Eine Warte-Phase ist immer acht Takte lang und es können weitere Warte-Phasen folgen. Das Folgen weiterer Warte-Phasen kündigt der SPI-Slave



an, in dem er die MISO-Signalleitung auf den Pegel „low“ treibt. Durch Anheben des Pegels auf „high“ im letzten Takt, wird die Warte-Phase beendet und mit der Phase 6 fortgefahren.

## 6. DATA-Phase

Die Phase sechs stellt die Daten-Phase dar, in der die Daten gelesen bzw. geschrieben werden. Ihre Länge ist abhängig von der in Phase drei gesetzten Anzahl der zu übertragenden Bytes.

Bei dem in **Abbildung 12** gezeigten Signalverlauf ist zu erkennen, dass am Anfang die Taktfrequenz im Ruhemodus Null beträgt. Mit dem Setzen der CS# Signalleitung am Eingang des SPI-Slave, wird die Kommunikation gestartet. Der SPI-Master beginnt mit der Erzeugung einer Taktfrequenz. Der TPM Bus-Takt ist bei SPI auf den Standard 24 MHz festgelegt. Er ist allerdings so angelegt worden, dass das TPM in einem breiten Bereich unterschiedlicher Taktfrequenzen arbeiten kann. So kann in Computer-Umgebungen mit niedriger Leistung die Taktfrequenz auf bis zu 14,3 MHz abgesenkt werden. In Umgebungen mit hoher Leistung kann dagegen die Taktfrequenz auf bis zu 66 MHz angehoben werden. Hersteller von TPM sind dazu angehalten, ihre Module für ein breites Spektrum an Taktfrequenzen auszulegen. Der Abbildung ist weiterhin zu entnehmen, dass die Taktfrequenz im Ruhemodus auf dem „low“ Pegel gehalten wird. Für TPMs ist nur der SPI-Modus 0 (CPOL=0, CPHA=0) zulässig. Gelesen werden demnach die Daten auf der steigenden Flanke des Taktes. Beim Lesen der übertragenden Bits muss beachtet werden, dass diese in msb übertragen werden. Es handelt sich um eine Lese-Transaktion (0), bei der ein Byte (00 0000) gelesen werden soll. Die Zieladresse dieser Transaktion lautet 0xD40024. Zieladressen des TPM beginnen nach TCG Spezifikationen über SPI mit einem Offset von 0xD4, so dass die 16-Bit Zieladresse auf 0x0024 zeigt. Anschließend folgt eine Warte-Phase, welche nicht verlängert werden muss (0000 0001). Das TPM antwortete mit dem angeforderten Byte 0x2C (0010 1100). [5, S. 127,134]

## 2.4 Bewertung der Schwachstellen

Ausgehend von den in 2.3 vorgestellten theoretischen Angriffspunkten soll hier eine Bewertung und die Praxistauglichkeit der Angriffe durchgeführt werden.

### 2.4.1 Brute-Force gegen Wiederherstellungsschlüssel

Der Vorteil dieses Angriffes ist, dass er immer durchgeführt werden kann, sofern man über ein Abbild der BitLocker-Partition besitzt. Dem ist sich auch Microsoft bewusst, weshalb der Wiederherstellungsschlüssel mit einer Länge von 48-Stellen ein sehr langes Passwort darstellt. Der Schlüsselraum für einen Wiederherstellungsschlüssel kann weiter eingegrenzt werden, da für diesen folgende Regeln gelten: [43]

- Jeder Block aus sechs Zahlen muss durch 11 ohne Rest teilbar sein
- Die Gesamtsumme eines Zahlenblocks muss kleiner gleich 720896 sein

Mit Hilfe dieser Information ergibt sich für den Schlüsselraum eine Größe von  $3,4 \cdot 10^{38}$ . Die Berechnung des Schlüsselraums wird in der Formel (1) durchgeführt.

$$\text{Schlüsselraum} = \left( \frac{\max(\text{Zahlenblock})}{11} \right)^{\text{Blöcke}} = \left( \frac{720896}{11} \right)^8 = 3,4 \cdot 10^{38} \quad (1)$$

In einer Versuchsreihe wurde ermittelt, wie viele Passwortkandidaten mit aktueller Hardware und dem Programm „John The Ripper“ pro Sekunde getestet werden können. Zusätzlich wurde berechnet, wie lange ein Austesten aller Passwortkandidaten dauern würde. Die Ergebnisse sind in der **Tabelle 11** dargestellt. Die detaillierten Ergebnisse dieser Messungen liegen als Anhang dieser Arbeit bei.

Hardware	Passwörter / Sekunde	Gesamtdauer [Jahre]
Nvidia RTX 3080 Ti	3263	$3,3069 \cdot 10^{27}$
Nvidia RTX 3080	3225	$3,3458 \cdot 10^{27}$
Nvidia RTX 2080 Ti	2167	$4,9794 \cdot 10^{27}$
Nvidia GTX 1080	1102	$9,7915 \cdot 10^{27}$
Nvidia GTX 1060	520	$20,7505 \cdot 10^{27}$
Nvidia GTX 750 Ti	171	$63,1010 \cdot 10^{27}$

**Tabelle 11:** John The Ripper – BitLocker Modus Performance

Die Zeit, die benötigt wird, ist selbst mit aktueller Grafikkartenunterstützung zu hoch. Mit einer Nvidia RTX 3080 Ti müssten im schlechtesten Fall die nächsten  $3,3069 \cdot 10^{27}$  Jahre alle Schlüssel getestet werden, um den richtigen Wiederherstellungsschlüssel zu finden. Zum Vergleich, das Alter unseres Universums wird gemäß der Urknall-Theorie lediglich auf  $1,4 \cdot 10^{10}$  Jahre geschätzt. [44, S. 454-457]

Der Angriff gegen den Wiederherstellungsschlüssel stellt zwar einen theoretischen Ansatz dar, der in der Praxis ohne weitere Eingrenzung des Schlüsselraumes aber nicht umsetzbar ist.

### 2.4.2 Cold Boot Angriff

Der ursprüngliche Cold Boot Angriff, der von der Princeton University durchgeführt wurde, bezog sich lediglich auf SDRAM, DDR und DDR2 Arbeitsspeicher. In aktuellen Computersystemen befindet sich der DDR-Speicher allerdings bereits in der fünften Generation. Mit der Einführung neuer Prozessoren und Arbeitsspeicher Versionen wurden auch neue Sicherheitselemente integriert, welche einen Cold Boot Angriff deutlich erschwert haben. Auf diese soll nachfolgend eingegangen werden.

Der Cold Boot Angriff auf ältere Arbeitsspeicher Generationen setzt voraus, dass die Daten unverschlüsselt auf dem Arbeitsspeicher abgelegt sind. Intel hat seit Einführung der zweiten Generation der Intel Core Prozessorfamilie das Daten

Scrambling eingeführt. Dies soll dafür sorgen, dass es zu keiner Überbeanspruchung der Daten Bus-Leitungen kommen kann. Werden viele Bits mit dem Wert „Eins“ in den Arbeitsspeicher geschrieben, so muss der Pegel der Datenleitung für einen längeren Zeitraum auf „high“ gehalten werden. Die elektrische Energie, die dabei auf den Datenbus einwirkt, ist definiert aus dem Produkt der Spannung, dem Strom und der Zeit. Eine hohe Energieeinwirkung kann den Datenbus beschädigen. Unter Umständen müssen aber mehrere Bits mit dem Wert „Eins“ hintereinander in den Arbeitsspeicher geschrieben werden. Intel hat das Problem gelöst, indem sie die Daten vor dem Schreiben mit Hilfe eines Scrambling Seeds Pseudo randomisieren. Die zuvor lange Abfolge von „Einsen“ wird in mehreren „Nullen“ und „Einsen“ aufgebrochen, was die Energieeinwirkung verringert. [45, S. 23] Diese Funktion, ursprünglich als Schutz vor zu hoher Energieeinwirkung, hat als Nebeneffekt, dass die Daten nicht im Klartext vorliegen und nur noch bei Kenntnis über den Scrambling Seed entschlüsselt werden können. Der Scrambling Seed verändert sich mit jedem neuen Systemstart. Wird der Arbeitsspeicher nun mittels Cold Boot und Umsetzen in ein neues System angegriffen, so können die Daten nur noch verschlüsselt ausgelesen werden. AMD Prozessoren verfügen über eine ähnliche Funktion, wobei dies als „AMD Memory Guard“ bezeichnet wird. [46] An der Friedrich-Alexander-Universität Erlangen-Nürnberg wurde in der wissenschaftlichen Arbeit „Lest we forget: Cold-boot attacks on scrambled DDR3 memory“ [35] der Cold Boot Angriff gegen ein DDR3 Arbeitsspeicher mit Scrambling durchgeführt.

Anstatt bei einem Cold Boot Angriff den Arbeitsspeicher auf ein anderes System umzusetzen, besteht jedoch weiterhin die Möglichkeit, mittels angepassten Betriebssystems den Zielrechner nach einem Reset zu starten. Nach einem Reset ändert sich der Scrambling Seed nicht und das System kann dann ebenfalls über den Scramble Seed ein Abbild der Daten aus dem Arbeitsspeicher unverschlüsselt erstellen. Wie allerdings in 2.1.4.1 beschrieben wurde, schützt das TPM gegen diese Art von Angriffen durch das Setzen eines MOR-Bit. Ein Auslesen des FVEK ist ohne weiteres deshalb nicht mehr möglich.

### 2.4.3 DMA-Angriff

Bei all den Vorteilen die DMA mit sich gebracht hat, ermöglichte es allerdings auch neue Angriffe, um den Arbeitsspeicher eines Systems auszulesen und zu manipulieren. Um solche Angriffe zu verhindern, werden von den Herstellern unterschiedliche Ansätze verfolgt, wobei sich als besten Schutz gegen DMA-Angriffe die „I/O memory management unit“ (IOMMU) bewährt hat.

Bei der IOMMU handelt es sich um eine eigene Hardwarekomponente, die zwischen Peripherie-Gerät und Arbeitsspeicher sitzt. Hauptbestandteil dieser Komponente ist das DMA-Remapping. Beim DMA-Remapping werden Speicheradressen, die über DMA angesprochen werden sollen, in alternative virtuelle Adressen übersetzt. Dadurch ist es einem Gerät nur möglich, auf für ihn beschränkte virtuelle Speicheradressen, aber nie direkt auf alle physikalische Speicheradressen zugreifen zu können. Das DMA-Remapping basiert auf mehreren Ebenen und arbeitet mit Übersetzungstabellen. Die Geräte sind in Domains organisiert und jede Domain verfügt über seine eigene IOMMU Konfiguration. Das bedeutet, dass mehrere Geräte derselben Domain sich die gleiche Speicherzuordnung teilen, aber nicht auf die Speicherzuordnung anderer Domains zugreifen können. [47] Nutzt ein System die Funktionalität der IOMMU, werden DMA-Angriffe wie der PCILeech Angriff wirkungslos. [33]

Des Weiteren wird bei einem DMA-Angriff vorausgesetzt, dass das System über entsprechende Schnittstellen verfügt. Diese Schnittstellen müssen das Hinzufügen weiterer Peripherie im laufenden Betrieb ermöglichen. Das Hinzufügen kompromittierter Hardware vor dem Systemstart wird unter Umständen durch die Integritätsmessung erkannt und verhindert ein booten des Betriebssystems.

### 2.4.4 TPM Bus-Sniffing

Das TPM Bus-Sniffing stellt einen relativ neuen Angriff dar, zu dem bisher wenige Erfahrungen und Informationen existieren. Das Mitlesen der Kommunikation zwischen TPM und System setzt voraus, dass das Gerät über ein diskretes TPM

verfügt und ein physikalischer Zugang zu diesem besteht. Über diesen können die Signalleitungen abgegriffen und ausgewertet werden.

Für den Angriff muss lediglich der Startvorgang des Betriebssystems aufgenommen und analysiert werden.

### 2.4.5 Zusammenfassung

In der **Tabelle 12** sind die hier vorgestellten Angriffe gegenübergestellt. Dabei werden Zeitaufwand und Voraussetzungen für den Angriff verglichen.

Der Brute-Force Angriff gegen den Wiederherstellungsschlüssel einer BitLocker verschlüsselten Partition hat keine weiteren Voraussetzungen und ist immer möglich. Wie in 2.4.1 gezeigt dauert ein solcher Angriff allerdings zu lange, um ihn in der Praxis durchführen zu können.

Gegen Angriffe zum Auslesen des FVEK aus dem Arbeitsspeichers existieren effektive Gegenmaßnahmen. Diese werden bereits vom Hardware-Hersteller implementiert oder durch BitLocker softwareseitig bereitgestellt. Die Voraussetzungen, die für den Cold-Boot und dem DMA-Angriff gegeben sein müssen, schränken die Zielgruppe für einen erfolgreichen Angriff stark ein.

Ein bisher weitestgehend unerforschter Angriff stellt das TPM Bus-Sniffing dar. Bei diesem gilt, dass lediglich ein diskretes TPM auf dem System vorhanden sein muss, um den Angriff durchführen zu können.

Von den hier vorgestellten Angriffen stellt das TPM Bus-Sniffing den Angriff mit den größten Erfolgchancen dar, weil dieser nur wenige Voraussetzungen besitzt und innerhalb weniger Minuten durchgeführt werden kann. Im weiteren Verlauf dieser Arbeit soll deshalb der TPM Bus-Sniffing Angriff untersucht werden. Ob und welche Randbedingungen Einfluss auf diesen Angriff haben könnten, werden im Kapitel 2 dieser Arbeit erforscht.

Angriff	Zeitaufwand	Voraussetzungen
Brute-Force	Jahre	<ul style="list-style-type: none"><li>• Keine</li></ul>
Cold-Boot	Minuten	<ul style="list-style-type: none"><li>• Arbeitsspeicher nicht verschlüsselt</li><li>• Arbeitsspeicher darf nicht überschrieben werden bei Neustart</li></ul>
DMA	Minuten	<ul style="list-style-type: none"><li>• Kein aktives IOMMU.</li><li>• Schnittstellen für Angriffe müssen gegeben sein.</li></ul>
Bus-Sniffing	Minuten	<ul style="list-style-type: none"><li>• Diskretes TPM</li></ul>

**Tabelle 12:** Gegenüberstellung der Angriffsvektoren

## **3 Praktische Validierung**

### **3.1 Vorbereitungen**

Für die Vorbereitungen des Angriffes muss ein Szenario festgelegt werden. Auf diesem wird die praktische Validierung des TPM Bus-Sniffing Angriffes durchgeführt. Die anzugreifenden Systeme werden vorgestellt und die Rahmenbedingungen für den Angriff festgelegt. Die speziell benötigte Hardware sowie deren Spezifikationen werden im Abschnitt 3.1.4 ermittelt. Zusätzlich wird Software zur Durchführung des Angriffes benötigt, die zunächst programmiert und im Abschnitt 3.1.5 vorgestellt wird.

#### **3.1.1 Szenario**

Das Szenario dieser Arbeit beschreibt einen Privatanwender ohne tiefere Windows Betriebssystemkenntnisse. Er verfügt über ein System mit diskreten TPM, das in der Firmware aktiviert ist. Der Nutzer hat die Ersteinrichtung seines Betriebssystems durchgeführt. Bei der Einrichtung ist er den Empfehlungen von Windows gefolgt und hat alle Einstellungen im Standard belassen. Nach dem ersten erfolgreichen Einloggen in das System hat der Benutzer das System auf den aktuellen Stand gebracht, indem er alle Sicherheitsrelevanten Updates eingespielt hat.



Für die Betriebssysteme Vista bis Windows 8.1 wurde im Anschluss BitLocker auf die Betriebssystem Partition aktiviert. Der Wiederherstellungsschlüssel wurde ausgedruckt und sicher verwahrt. Ab Windows 10 hat der Nutzer mit der Ersteinrichtung ein Microsoft Online Konto eingerichtet. Aufgrund des vorhandenen TPM auf dem System wurde damit BitLocker automatisch auf dem System aktiviert. Der Wiederherstellungsschlüssel wurde in seiner OneDrive Cloud gesichert. Ein potenzieller Angreifer hat keinen Zugriff auf den Wiederherstellungsschlüssel in gedruckter Form oder auf die Cloud des Nutzers.

### 3.1.2 Zielsysteme

Als Zielsysteme kommen nur jene in Betracht, die über ein diskretes TPM verfügen. Bei der Auswahl wird darauf geachtet, dass eine Variation von TPMs vorhanden ist. Das bedeutet, dass sowohl TPMs verschiedener Hersteller, Versionen und Kommunikationsprotokollen vertreten sind. Die Auswahl ist auf drei Geräte gefallen, welche nachfolgend vorgestellt werden.

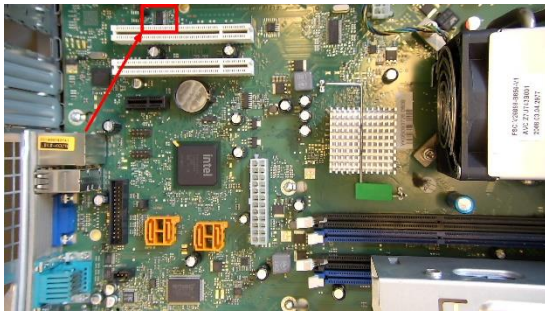
#### 3.1.2.1 Fujitsu Siemens Esprimo E5720



**Abbildung 13:** Fujitsu Siemens Esprimo E5720

Der Fujitsu Siemens Esprimo E5720 ist ein Small Formfaktor PC, der von der Firma Fujitsu Siemens ab 2008 vertrieben wurde. Er ist ausgestattet mit einem Intel Pentium Dual CPU E2160. [48] Um das System auch für neuere

Betriebssysteme vorzubereiten, wurde der Arbeitsspeicher des Gerätes auf 16GB DDR2 erweitert, die bestehende 80GB HDD gegen eine 256 GB SSD getauscht und ein BluRay Brenner hinzugefügt. Das System verfügt über ein BIOS-Firmware von Phoenix Technologies Ltd. in der Version 6.00 R1.182594A1.



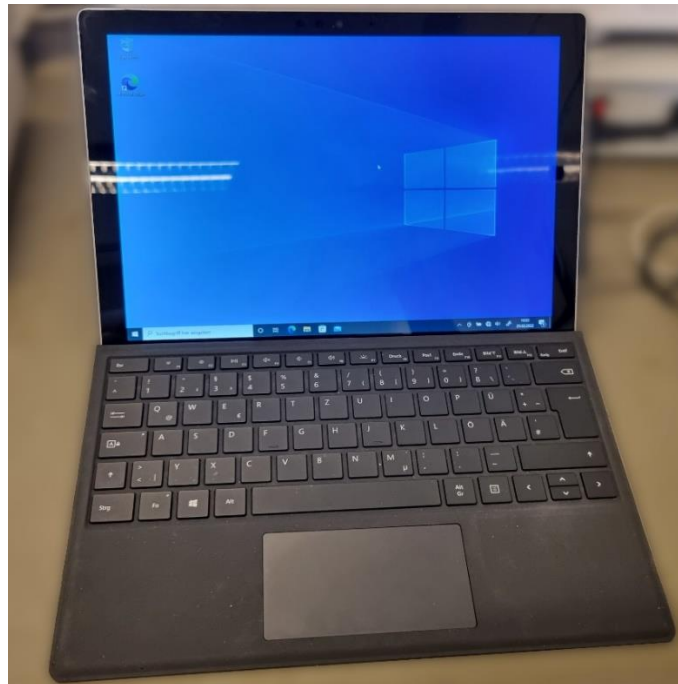
**Abbildung 14:** Mainboard D2594



**Abbildung 15:** Infineon SLB9635TT12

In dem Gerät befindet sich ein diskretes TPM (SLB9635TT12), welches auf dem Mainboard verlötet ist. Dabei handelt es sich um ein TPM 1.2 der Firma Infineon, das über den LPC-Bus kommuniziert.

### 3.1.2.2 Microsoft Surface Pro 5 (Model 1807)

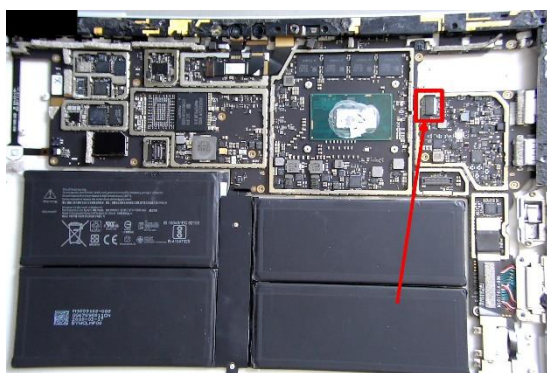


**Abbildung 16:** Microsoft Surface Pro 5

Das Microsoft Surface Pro 5 wurde im Mai 2017 veröffentlicht und stellt den Nachfolger des Microsoft Surface Pro 4 dar. Es wird standardmäßig mit Windows 10 Pro ausgeliefert. Das Gerät verfügt über einen 256 GB internen Speicher und 8 GB RAM. Als Prozessor kommt ein Intel Core i5-Prozessor der 7. Generation zum Einsatz (Core i5-7300U). Wie schon seine Vorgängermodelle, ist das Microsoft Surface Pro 5 ebenfalls mit einem diskreten TPM bestückt. [49] Das System besitzt eine UEFI-Firmware von Microsoft in der Version 235.3732.768. Die Option Secure Boot ist auf diesem Gerät eingeschaltet.

Bei Secure Boot handelt es sich um eine UEFI-Spezifikation, die dazu dient, die Integrität der Firmware, dem Bootloader, dem Betriebssystem-Kernel und den Betriebssystem-Treibern sicherzustellen. Dies erfolgt über kryptografische Signaturen, die beim Startprozess von Secure Boot geprüft werden. Verfügt das Betriebssystem beispielsweise über keine gültige Signatur, wird ein Starten des Systems unterbunden. Secure Boot verhindert dadurch das Starten veränderter oder bösartiger Software. [50]

Bei dem TPM handelt es sich um ein TPM 2.0 der Firma Nuvoton (NPCT650SBCWX). Der Chip ist verlötet und in einem 28-Pin TSSOP Package untergebracht. Er kommuniziert über den LPC-Bus mit dem System. In der **Abbildung 17** und **Abbildung 18** sind das Mainboard und das TPM des Microsoft Surface Pro 5 dargestellt.

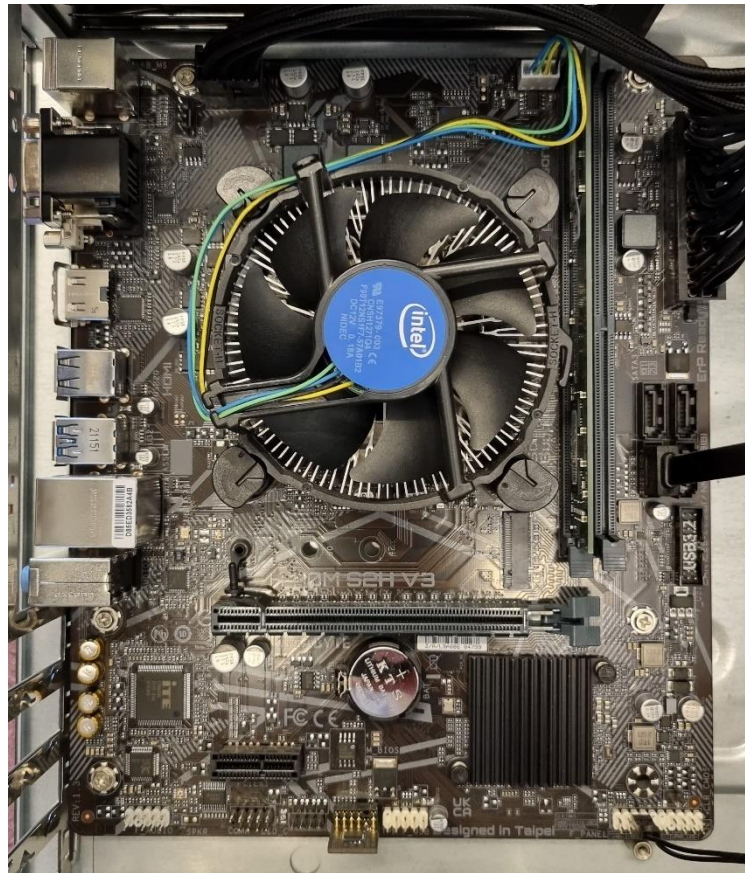


**Abbildung 17:** Mainboard Surface Pro 5



**Abbildung 18:** Nuvoton NPCT650

### 3.1.2.3 Gigabyte H410M S2H V3 mit GC-TPM2.0 SPI 2.0

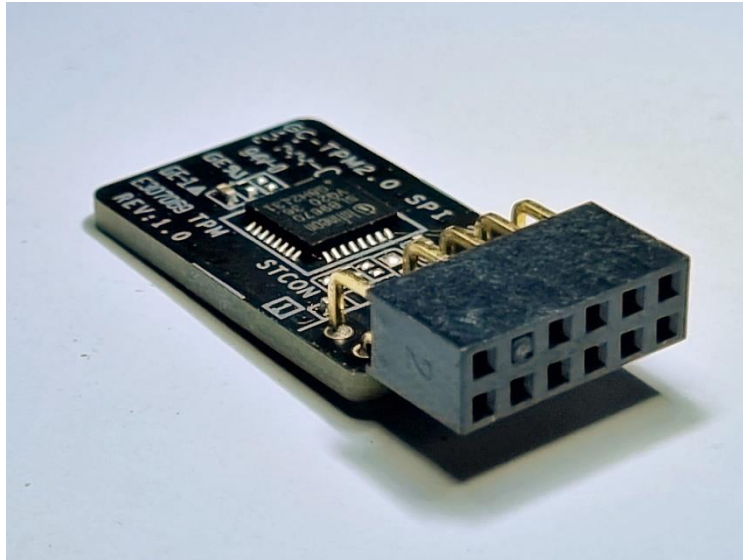


**Abbildung 19:** Gigabyte H410M S2H V3

Für das dritte Untersuchungsobjekt wird ein eigenes System aus aktueller Hardware aufgebaut. Dieses ist bestückt mit einem Gigabyte H410M S2H V3 Mainboard, einem Intel Celeron G5905 Prozessor, 4GB DDR4 Arbeitsspeicher und einer 2 TB großen SSD. Es verfügt über eine UEFI Firmware der Firma American Megatrends in der Version FC. Die Option Secure Boot ist auf diesem Gerät ausgeschaltet.

Das Mainboard verfügt zwar selbst über kein diskretes TPM, bietet aber eine Schnittstelle, um ein TPM Header Modul anschließen zu können. Als TPM Header Modul wurde das GC-TPM2.0 SPI 2.0 Modul gewählt. Dieses ist in **Abbildung 20** dargestellt.





**Abbildung 20:** GC-TPM2.0 SPI 2.0

Das Modul verfügt über ein TPM 2.0 (SLB9670VQ2.0) der Firma Infineon. Als Kommunikationsschnittstelle wird der SPI-Bus genutzt. Das TPM ist in einem 32-Pin QFN-Package untergebracht. Der Intel Celeron Prozessor verfügt über die Intel Plattform Trust Technology und somit über ein fTPM. Mit dem aufgesteckten TPM Header Modul verfügt das System sowohl über ein fTPM als auch ein dTPM. In einem System kann immer nur ein TPM aktiv sein. Daher wird in der Firmware des Mainboards explizit das dTPM ausgewählt. Darüber hinaus bietet die Firmware des Gigabyte Mainboards tieferegehende Einstellungen am TPM an. Diese werden nachfolgend kurz in der **Tabelle 13** vorgestellt.

Option	Beschreibung	Wert
Security Device Support	Aktiviert bzw. deaktiviert die Unterstützung für diskrete TPM	Enable
SHA-1 PCR Bank	Erlaubt bzw. verbietet das Nutzen von SHA-1 Hashes im PCR	Eingeschaltet
SHA256 PCR Bank	Erlaubt bzw. verbietet das Nutzen von SHA256 Hashes im PCR	Eingeschaltet

Pending operation	Wenn aktiviert, wird das TPM beim nächsten Neustart gelöscht	Ohne
Platform Hierachy	Aktiviert das Nutzen einer Plattform Hierarchie	Eingeschaltet
Storage Hierachy	Aktiviert das Nutzen einer Speicher Hierarchie	Eingeschaltet
Endorsement Hierachy	Aktiviert das Nutzen einer Endorsement Hierarchie	Eingeschaltet
TPM2.0 UEFI Spec Version	Legt fest mit welcher TCG Spezifikation das TPM betrieben werden soll. Für ältere Betriebssysteme kann das TPM auf die TCG Spezifikationen eines TPM 1.2 beschränkt werden. Neuere Systeme können mit den Spezifikationen eines TPM2.0 betrieben werden.	TCG_2
Physical Presence Spec Version	Festlegen der PPI Spezifikation Version 1.2/1.3. Einige ältere Betriebssysteme unterstützen nur den PPI Standard 1.2	1.3
Device Select	Festlegen, ob das TPM als TPM1.2 oder TPM2.0 verwendet werden soll. Im Auto Modus wird das System mit einem TPM2.0 gestartet. Kann dieses nicht erkannt werden, startet es mit den Spezifikationen eines TPM1.2	Auto

**Tabelle 13:** TPM Firmware Einstellungen

Für den Versuch verbleiben die restlichen Einstellungen in den Standardwerten der Firmwarekonfiguration.

#### 3.1.2.4 Gegenüberstellung

In der **Tabelle 14** sind die ausgewählten Zielsysteme gegenübergestellt. Die Systeme wurden so gewählt, dass sowohl ein älteres TPM 1.2 und zwei neuere TPM 2.0 untersucht werden können. Hierbei unterscheiden sich die zwei TPM 2.0 in der Art ihres Kommunikation Protokolls. Es soll dabei untersucht werden, ob es Unterschiede zwischen dem LPC- und dem SPI-Bus bei der Kommunikation mit dem TPM gibt. Die TPM Hersteller Infineon und Nuvoton wurden ausgewählt, da diese die Marktführer in diesem Bereich sind. Die zu untersuchenden TPMs unterscheiden sich weiterhin in ihrer Bauform (28-TSSOP, 32-QFN) und in ihrer Ausführungsart (verlötet, Header Modul), um mögliche Unterschiede im Angriff identifizieren zu können.

Neben dem Einfluss des TPMs auf BitLocker und den Angriff, soll auch untersucht werden, welchen Einfluss die Firmware des Zielsystems, hat. Hierzu wurde mit dem Fujitsu Siemens Espriomo E5720 ein System mit BIOS und mit den beiden anderen Geräten Zielsysteme mit UEFI-Firmware gewählt.

Bei den beiden Geräten mit UEFI-Firmware soll untersucht werden, welchen Einfluss die Einstellung Secure Boot auf BitLocker und dem TPM-Bus-Sniffing Angriff besitzt.

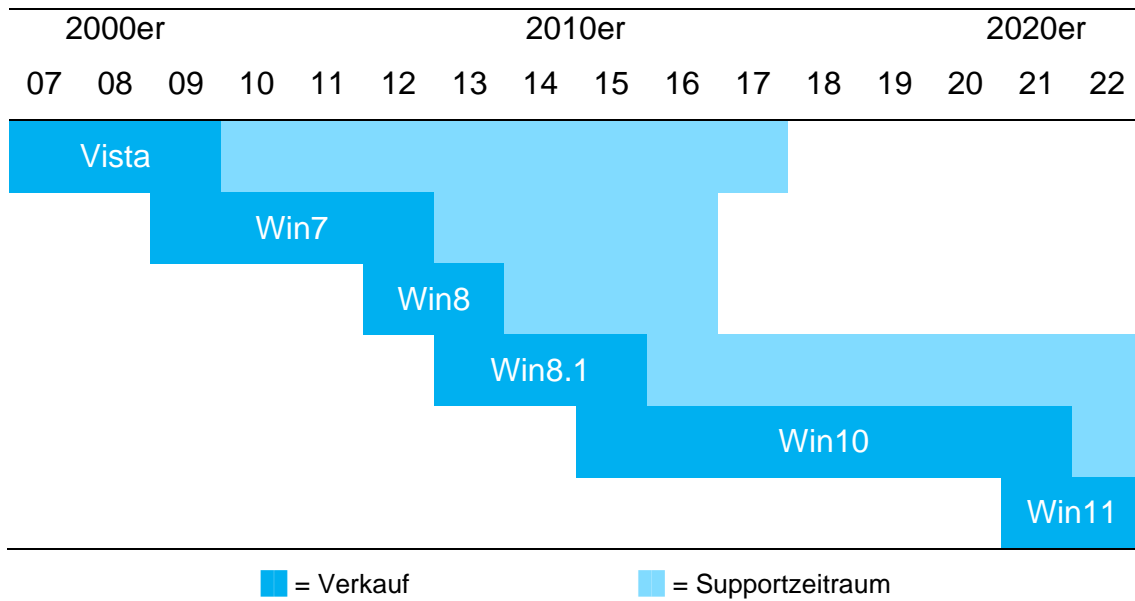
Merkmal	Fujitsu Siemens Esprimo E5720	Microsoft Surface Pro 5	Gigabyte H410M S2HV3
TPM	1.2	2.0	2.0
Protokoll	LPC	LPC	SPI
Hersteller	Infineon	Nuvoton	Infineon
IC	SLB9635TT12	NPCT650SBCWX	SLB9670VQ2.0
Package	28-TSSOP	28-TSSOP	32-Pin QFN
Art	verlötet	verlötet	Header Modul
Firmware	BIOS	UEFI	UEFI
Secure Boot	-	An	Aus

***Tabelle 14:** Gegenüberstellung Zielsysteme*

### 3.1.3 Betriebssysteme

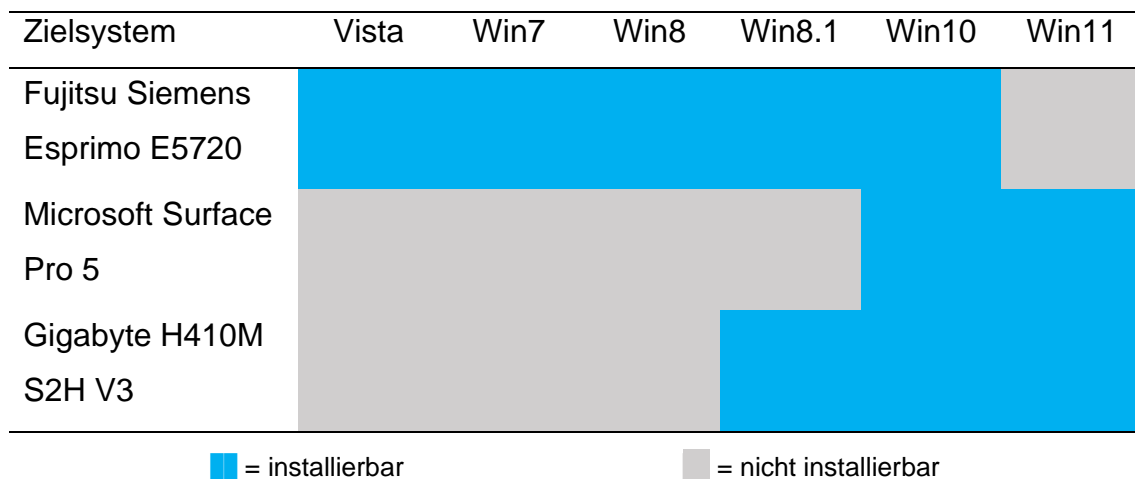
Neben dem Einfluss der Hardware auf den Bus-Sniffing Angriff, soll ebenso der Einfluss des Betriebssystems und seiner Komponenten untersucht werden. BitLocker wurde von Microsoft erstmals 2007 mit Windows Vista eingeführt. Seitdem ist es fester Bestandteil von Windows. BitLocker ist dabei ausschließlich den Versionen Enterprise und Pro bzw. Ultimate des Betriebssystems vorbehalten. In der **Tabelle 15** sind alle Windows Betriebssysteme aufgeführt, die für einen potenziellen BitLocker Bus-Sniffing Angriff in Frage kommen.





**Tabelle 15:** Windows Produkt Lebenszyklus [51] [52] [53] [54] [55] [56]

Bei den Versuchen werden die Betriebssysteme immer auf den aktuellen Stand, zum Zeitpunkt dieser Arbeit (Juli 2022), gebracht und alle Sicherheitsupdates installiert. Die Windows Betriebsversionen besitzen unterschiedliche Systemvoraussetzungen. Es wurde deshalb im Vorfeld geprüft, welche Windows Versionen auf den zuvor vorgestellten Zielsystemen installiert werden können. Das Ergebnis ist in **Tabelle 16** dargestellt.



**Tabelle 16:** Installierbare Betriebssysteme auf den Zielsystemen

Es sei darauf hingewiesen, dass alte Betriebssysteme ohne weitere Anpassung nicht auf moderner Hardware funktionieren. Der Grund dafür ist meist die fehlende Treiberunterstützung für CPU und Peripherie wie z.B. USB3.0. Bei dem Microsoft Surface Pro 5 werden für das Gerät ausschließlich Hardware Treiber für das Windows Betriebssystem Windows 10, ab Version 1703 und höher bereitgestellt. [57] Auch für das Gigabyte H410M System werden von Gigabyte nur Treiber ab Windows 8.1 bereitgestellt. [58] Es ist somit nicht ohne weiteres möglich, ältere Windows Betriebssysteme auf den Geräten zu betreiben. Um dies möglich zu machen, könnten Treiber unter Umständen zum Installationsmedium hinzugefügt werden. Für diesen Versuch wurde davon allerdings Abstand genommen. Es wurden nur Betriebssysteme installiert, die sich im Standard mit dem von Microsoft vorgegeben Installationsmedium installieren ließen.

Im Fall des Fujitsu Siemens Esprimo E5720 kann kein Windows 11 installiert werden. Hier sind nicht fehlende Treiber, sondern das TPM in der Spezifikation 1.2 ausschlaggebend. Windows 11 setzt das Vorhandensein eines neueren TPM2.0 auf dem Gerät voraus. [59]

BitLocker bietet die Möglichkeit den Verschlüsselungsalgorithmus und die VMK Schutzmaßnahmen anzupassen. Das Verändern dieser Parameter erfordert allerdings tiefere administrative Veränderungen am System. Bei den Versuchsobjekten wird ausschließlich der im Standard gesetzte Algorithmus und VMK Schutz untersucht.

### **3.1.4 Randbedingungen Logikanalysator**

Um den Datenverkehr auf den Busleitungen mitlesen zu können, wird ein Logikanalysator benötigt. Dieser misst in einer festen Frequenz den Spannungspegel auf einer Signalleitung. Der gemessene Wert wird entsprechend eines Schwellwertes einem booleschen Wahrheitswert „Null“ bzw. „Eins“ zugeordnet.

Um zu ermitteln welche Eigenschaften der Logikanalysator für diesen Versuch besitzen muss, wird in einer Messreihe die Frequenz des Bus-Takt gemessen.

Weiterhin wird ermittelt, wie lange die Startphase eines Windows Betriebssystems andauert und wie lange der Logikanalysator diesen Vorgang aufzeichnen können muss.

#### 3.1.4.1 Amplituden-/ Frequenzgang messen

Beim Messen von kontinuierlichen Signalen muss beachtet werden, dass das Nyquist-Shannon-Abtasttheorem (bzw. Whittaker-Kotelnikow-Shannon-Abtasttheorem) eingehalten wird. Dieses gilt für Tiefpass-Signale, wobei das Signal mit der höchsten Frequenz die Bandbreite bestimmt. Als höchste Frequenz kann in diesem Fall das Taktsignal des Busses angenommen werden. Das Abtasttheorem besagt:

„Ein kontinuierliches Signal der Bandbreite  $B$  kann aus seiner abgetasteten Version (Abtastfrequenz  $f_A$ ) nur dann fehlerfrei rekonstruiert werden, wenn  $f_A > 2B$  ist.“ [60]

Wird das Abtasttheorem nicht eingehalten, können Aliasing-Effekte auftreten, die zu Messfehlern führen. [60, S. 158-161]

Um die tatsächliche Taktfrequenz des Busses zu ermitteln, muss das Zeitsignal der jeweiligen Zielsysteme während der Startphase aufgenommen werden. Mit Hilfe der Fourier-Transformation kann aus dem Zeitsignal dann das Frequenzspektrum bestimmt werden. Aus dem Frequenzspektrum wiederum stellt die Frequenz mit dem größten Leistungsanteil die Bus-Taktfrequenz dar. Für die Ermittlung des Frequenzspektrums aus dem Zeitsignal wurde ein Skript in Python geschrieben, das als Anhang dieser Arbeit beiliegt.

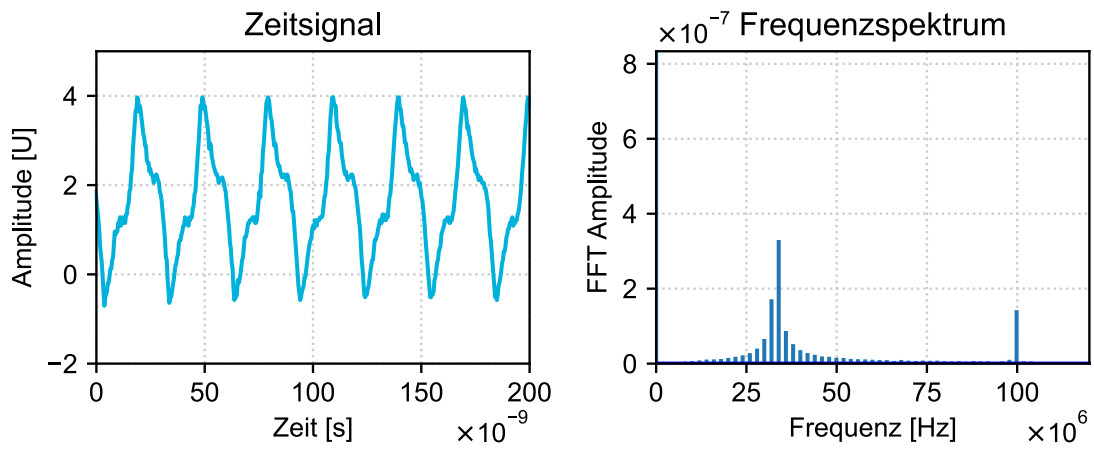
In der **Abbildung 21** sind die Messungen des Bus-Taktes zu den jeweiligen Zielsystemen dargestellt. Die Frequenz des Bus-Taktes ist bei den Geräten unterschiedlich. Nur der Bus Takt des Fujitsu Siemens Esprimo E5720 hält sich an die in 2.3.3.1 beschriebene Spezifikation von 33 MHz. Beim Microsoft Surface Pro 5 beträgt die Taktfrequenz des LPC-Bus ca. 23,8 MHz und die des Gigabyte SPI-Bus lediglich 14,3 MHz. Die Gründe für die Abweichungen, von den zu erwartenden 33 MHz, liegen in den Energiesparoptionen der Zielsysteme. Diese

ermöglichen es dem System die Taktfrequenz variabel zu verändern und so Energie zu sparen. In der **Abbildung 22** wird der Einfluss des Betriebssystems auf den Bus-Takt des Fujitsu Siemens Esprimo E5720 untersucht. Dabei kann festgestellt werden, dass im Zeitsignal minimale Unterschiede auftreten. Einen Einfluss des Betriebssystems auf die Taktfrequenz kann allerdings nicht festgestellt werden. Diese beträgt im Fall des Fujitsu Siemens Esprimo E5720 immer 33 MHz.

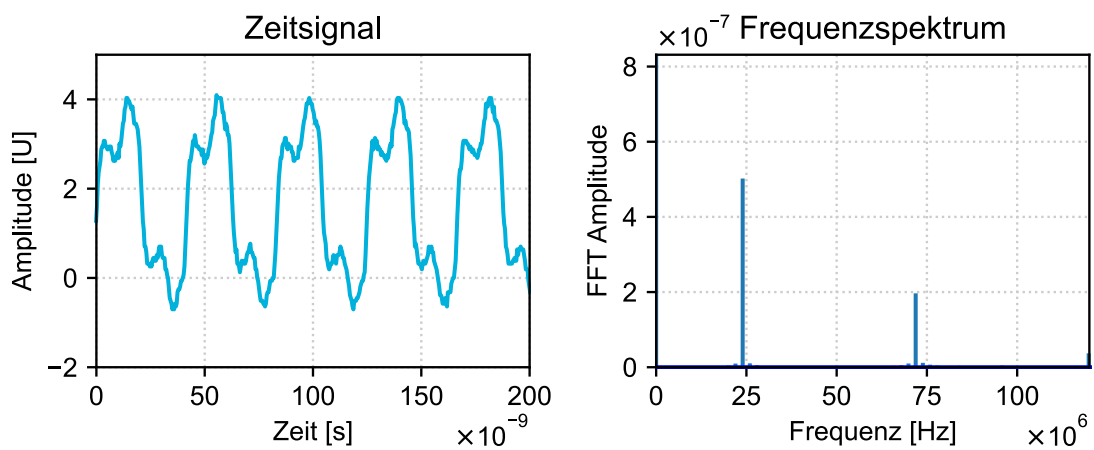
Als Randbedingung für den Logikanalysator ergibt sich für die Abtasttiefe somit eine Frequenz von mindestens:

$$f_{Abt} > 2 \cdot 33 \text{ MHz} = 66 \text{ MHz} \quad (2)$$

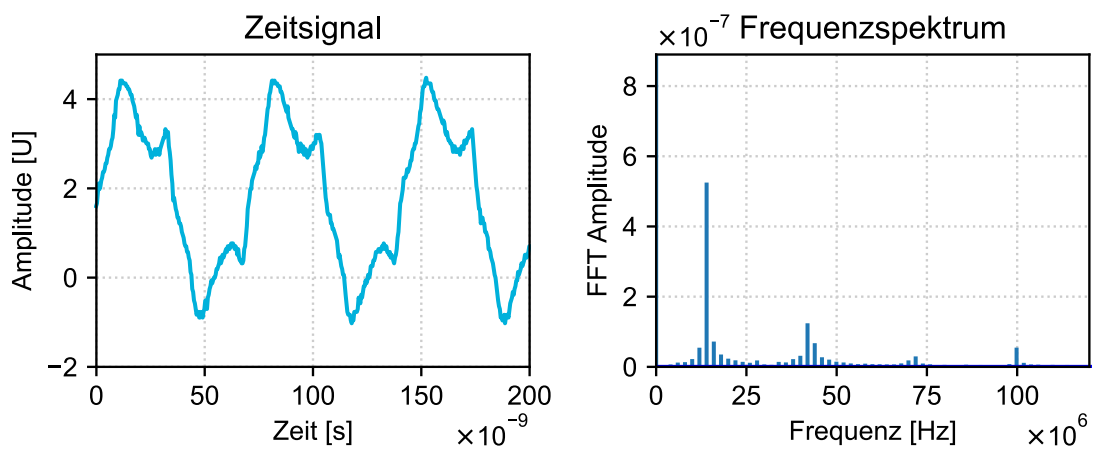
## Esprimo E5720 - Windows Vista



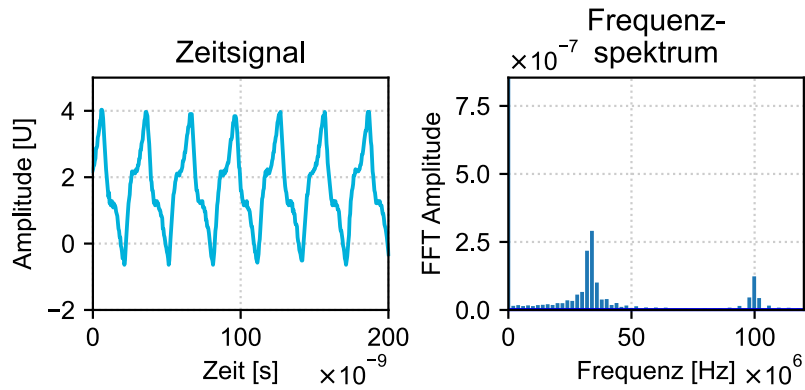
## Surface Pro 5 - Windows 11



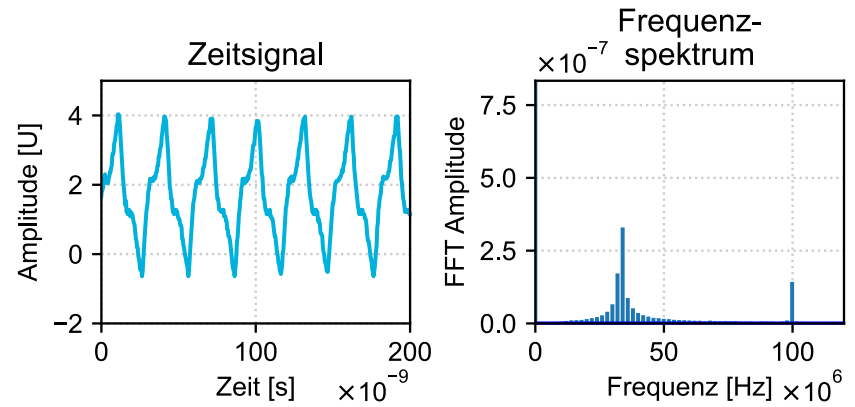
## Gigabyte H410M - Windows 10

**Abbildung 21:** Zeitsignal und Frequenzspektrum des Bus-Taktes

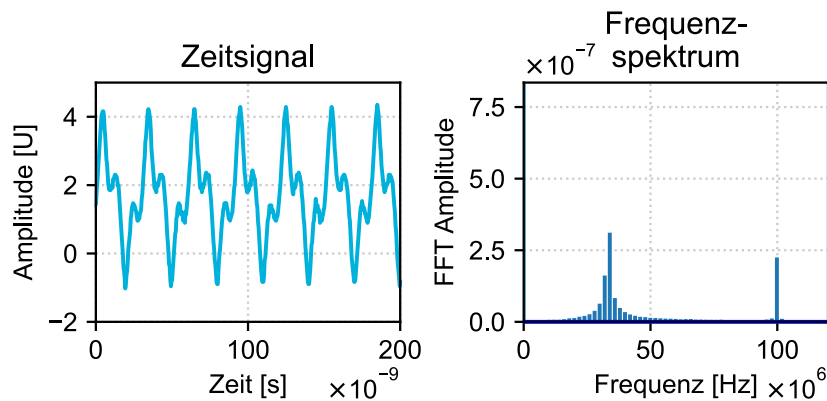
Esprimo E5720 - Windows 7



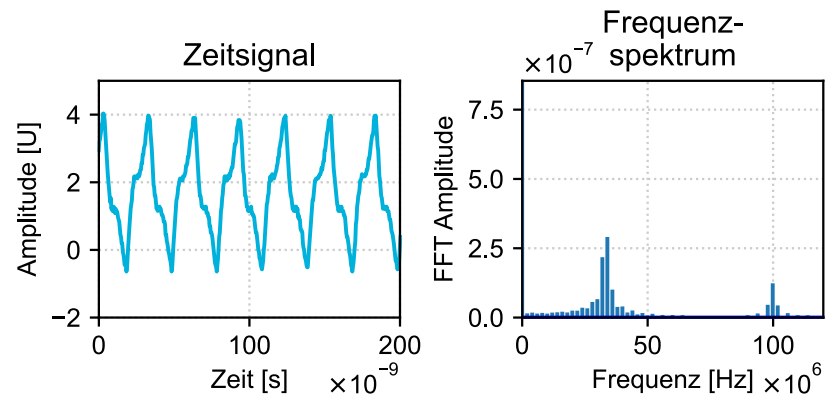
Esprimo E5720 - Windows 8



Esprimo E5720 - Windows 8.1



Esprimo E5720 - Windows 10



**Abbildung 22:** Einfluss des Betriebssystem auf den Bus-Takt

### 3.1.4.2 Bootdauer messen

Da nicht ersichtlich ist, zu welchem Zeitpunkt eine Kommunikation zwischen dem System und dem TPM stattfindet, ist es erforderlich, dass der gesamte Boot Prozess des Betriebssystems aufgezeichnet wird.

Ein Logikanalysator verfügt in der Regel über einen festen Speicher, in dem die gemessenen Signale zwischengespeichert werden. Entsprechend der Abtasttiefe und der Länge der Aufnahme besteht die Gefahr, dass dieser Speicher vollläuft bevor der Windows Boot Prozess abgeschlossen ist. Aus diesem Grund soll die Zeit zum Starten des Betriebssystems ermittelt werden, um die notwendige Größe des internen Speichers bestimmen zu können.

Zum Messen der Startdauer der unterschiedlichen Windows Betriebssysteme, auf den jeweiligen Zielsystemen, wird auf den Windows-eigenen Diagnostics-Performance Bericht zurückgegriffen. Bei jedem erfolgreichen Systemstart speichert Windows in die Event-Logs einen Eintrag mit der ID 100. Dieser beinhaltet die Zeitspanne, die Windows zum Starten des Betriebssystems benötigt hat. Der Eintrag dient als Systemstart-Leistungsüberwachung um zu ermitteln, welche Prozesse unter Umständen das System ausbremsen.

Für diesen Versuch wurden die jeweiligen Systeme fünf Mal gestartet. Aus den aufgenommenen Messwerten wird dann der Mittelwert bestimmt. In der **Tabelle 17** sind die Ergebnisse der Messungen zu den jeweiligen Betriebs- und Zielsystemen aufgeführt.

Zielsystem	Vista [ms]	Win7 [ms]	Win8 [ms]	Win8.1 [ms]	Win10 [ms]	Win11 [ms]
Fujitsu Siemens Esprimo E5720	39550,4	25765,2	20973,2	52799,2	84293,4	
Microsoft Surface Pro 5					36465,6	33617,6
Gigabyte H410M S2H V3				30746,8	24767,2	50234,8

**Tabelle 17:** Systemstartzeit

Die Zeiten sind in Millisekunden angegeben. Den längsten Systemstart besitzt die Kombination aus Fujitsu Siemens Esprimo E5720 und Windows 10. Hier dauerte ein vollständiger Startprozess 84 Sekunden. Der Logikanalysator muss in der Lage sein, mit entsprechender Abtasttiefe die volle Zeit aufnehmen zu können. Die Ergebnisse der einzelnen Messungen befinden sich im Anhang dieser Arbeit.

### 3.1.5 Dekodier-Software (ARNE)



```
+-----+
|                                             |
|  ARNE  |
|                                             |
|  (c) Sebastian Lasogga  |
|                                             |
+-----+

usage: ARNE -i INPUT [-o OUTPUT] -k KANAL [-d DECODER] [-
h] [-nt NOTIME]
ARNE: error: the following arguments are required: -i/--
input, -k/--kanal
```

**Abbildung 23:** Programmstart ARNE

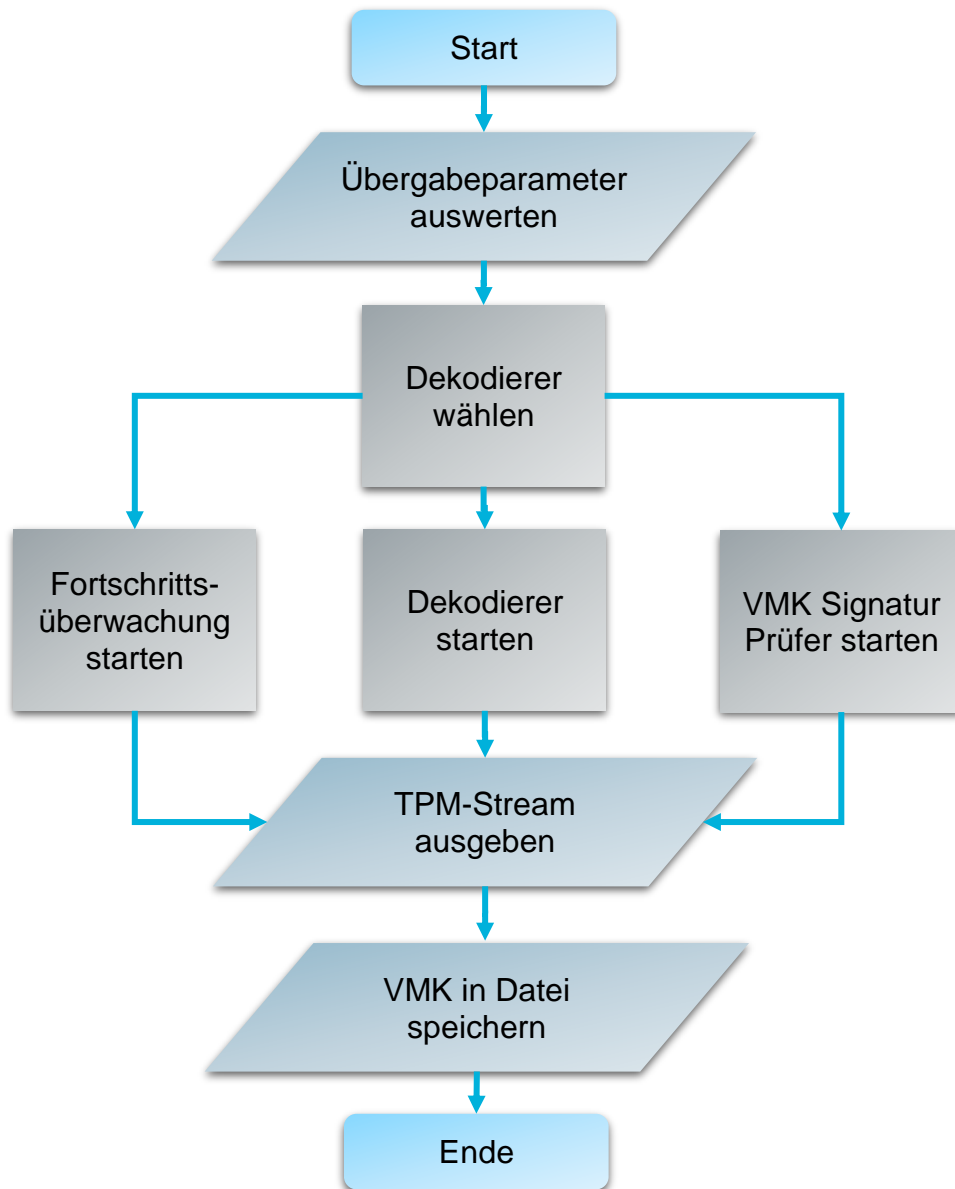
Es kann angenommen werden, dass beim Windows Startprozess mehrere tausend Pakete zwischen Host und TPM ausgetauscht werden. Jedes Paket manuell zu analysieren ist sehr zeitaufwendig und birgt die Gefahr, dass beim Dekodieren Fehler unterlaufen. Um dies zu unterbinden soll, als Teil dieser Arbeit, ein Programm entworfen werden, das in der Lage ist, Bus-Pakete zu erkennen, zu dekodieren und nach dem entsiegelten VMK zu untersuchen. Wenn der entsiegelte VMK in der Kommunikation gefunden werden konnte, dann soll dieser als Datei abgespeichert werden.

Das Programm dieser Arbeit trägt den Namen „ARNE“ (Advanced Re-Engineering of Network-Interception Encryption) und ist in Python 3 geschrieben. Es besteht aus den vier Teilen „*main.py*“, „*var.py*“, „*lpc\_decoder.py*“ und



„*spi\_decoder.py*“. Die Skripte und deren Funktionsweise werden nachfolgend beschrieben. Der vollständige Programmcode ist als Anhang dieser Arbeit beigefügt.

### 3.1.5.1 *main.py*



**Abbildung 24.** Ablaufdiagramm *main.py*

Das Skript „*main.py*“ stellt den Startpunkt des Programms dar. In ihm werden die notwendigen Bibliotheken und andere Programmteile eingebunden. (Zeile 1-10)

Für die ordnungsgemäße Ausführung des Programms werden folgende Bibliotheken benötigt:

- sys
- threading
- time
- argparse (>= 1.1)
- re (>= 2.2.1)
- binascii

Das Programm kann mit mehreren Parametern aufgerufen werden, wobei „*main.py*“ die Auswertung der Übergabeparameter übernimmt. (Zeile 90-141) Dabei kann es selbstständig erkennen, ob alle notwendigen Parameter übergeben wurden und entsprechend der Übergabeparameter den Bus-Typ bestimmen.

Sind alle notwendigen Parameter beim Programmstart übergeben worden, beginnt das Programm damit, die Anzahl der Messpunkte in der Übergabedatei zu ermitteln (Zeile 151). Dies ist für die spätere Fortschrittserkennung notwendig.

Anschließend wird entsprechend des Bus-Typs der jeweilige Bus-Dekodierer als neuer Thread angelegt. Das Anlegen von Threads ermöglicht die parallele Verarbeitung mehrerer Aufgaben zur gleichen Zeit. (Zeile 148-155)

```
181 # Wähle Dekoder
182 if(decoder == "LPC"):
183     thread1 = Thread( target=lpc_decoder.func,
184                      args=(filename,kanal,spalten_offset, sr) )
185 elif(decoder == "SPI"):
186     thread1 = Thread( target=spi_decoder.func,
187                      args=(filename,kanal,spalten_offset, sr) )
188 else:
189     print("Fehler!")
190     sys.exit()
```

**Abbildung 25:** *main.py*

Während der Bus-Dekodierer die Eingabedatei analysiert, startet „*main.py*“ zusätzlich zwei weitere Threads. Ein Thread für die Fortschrittsanzeige, die

anhand der Gesamtzahl der Messpunkte und der bereits untersuchten Messpunkte den Fortschritt des Bus-Dekodierers anzeigt (Zeile 19-26). Ein weiterer Thread wird für ein Kontrollprogramm gestartet, das in dem bereits gefundenen TPM-Stream nach der Signatur des entsiegelten VMKs sucht. (Zeile 28-44) Dabei handelt es sich um die Signatur, die im Abschnitt 2.2.2 für den entsiegelten VMK vorgestellt wurde.

```
22 print("+ Fortschritt: " +  
    str(int(var.zeile*100/zeile_anzahl)) + " %, TPM  
    Frames: " + str(int(len(var.stream)/2)), end="\r")
```

**Abbildung 26:** main.py (Zeile 22)

```
33 vmk_sig_match = re.search("2c000[0-6]000[1-9]000[0-  
    1]000[0-5]200000", var.stream)
```

**Abbildung 27:** main.py (Zeile 33)

Ist der Bus-Dekodierer durchgelaufen oder hat das Kontrollprogramm vorzeitig einen entsiegelten VMK entdeckt, werden alle drei Threads abgebrochen und „main.py“ wird fortgesetzt.

Es folgt die Ausgabe des dekodierten TPM-Stream. Wurde ein entsiegelter VMK im TPM-Stream gefunden und wurde als Übergabeparameter eine Ausgabedatei angegeben, wird der Schlüssel in eine entsprechende Datei gespeichert. (Zeile 173-179)

```
173 if(args['output']):  
174     try:  
175         with open(filename_o, 'wb') as f:  
176             f.write(binascii.unhexlify(var.vmk[0:64]))  
177             print("Der VMK wurde nach "+filename_o+"  
    abgespeichert")  
178     except:  
179         print("Fehler beim anlegen des VMK")
```

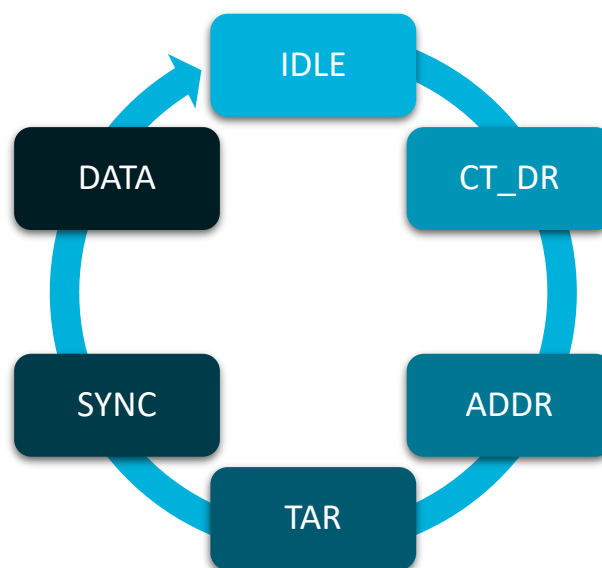
**Abbildung 28:** main.py (Zeile 173-179)

### 3.1.5.2 var.py

Das Skript „*var.py*“ dient dazu, globale Variablen zu definieren. Über die Import-Funktion von Python werden diese dann den anderen Skripten zur Verfügung gestellt.

### 3.1.5.3 lpc\_decoder.py

Das „*lpc\_decoder.py*“ Skript dekodiert die aufgenommenen Daten des LPC-Buses. Bei der Dekodierung orientiert sich das Skript an dem im Abschnitt 2.3.3.1 gezeigten Aufbau eines LPC-Frames. Das Skript stellt einen Zustandsautomaten dar, der auf den fallenden Flanken des Bus-Taktes die Werte der Signalleitungen interpretiert. In der **Abbildung 29** sind die möglichen Zustände des Zustandsautomaten dargestellt.



**Abbildung 29:** Zustandsautomat *lpc\_decoder.py*

In einer Schleife werden die Zeilen der Eingabedatei nacheinander eingelesen und die Werte den entsprechenden Signalleitungen zugeordnet. Nach dem Einlesen der Zeile erfolgt die Auswertung durch den Zustandsautomat.

Das Skript startet im Zustand IDLE und wartet so lange, bis auf einer fallenden Flanke des Bus-Taktes die LFRAME# Signalleitung auf „low“ gezogen wird. Tritt

dieses Ereignis ein, werden die Signalleitungen LAD0 bis LAD3 analysiert. Ergeben diese das Bitmuster für ein TPM-Frame, leert das Programm die Variablen aus vorherigen Aufnahmen und wechselt anschließend in die CT\_DR-Phase. (Zeile 55-74)

```
55 #IDLE Phase
56 elif(PHASE == "IDLE" and lframe == "0"):
57     idle.clear()
58     idle.append(lad3)
59     idle.append(lad2)
60     idle.append(lad1)
61     idle.append(lad0)
62
63     if(hex(int("".join(idle),2)) == "0x5"):
64         #Beginne neuen Zyklus und leere Variablen
        vorheriger
65         ct_d.clear()
66         addr.clear()
67         tar.clear()
68         sync.clear()
69         data.clear()
70
71         #Wechsel in die CT_D Phase
72         PHASE="CT_D"
73     else:
74         PHASE="IDLE"
```

**Abbildung 30:** *lpc\_decoder.py* (Zeile 55-74)

In der CT\_DR-Phase wartet das Skript auf die nächste fallende Bustakt-Flanke und prüft, ob es sich um einen Lesezyklus des LPC-Bus handelt. Sollte dies nicht der Fall sein, bricht der Zustandsautomat an dieser Stelle ab und wechselt zurück in die IDLE-Phase. Andernfalls wird in die ADDR-Phase gewechselt. (Zeile 78-88)

```
78 #CT_D Phase
79 if(PHASE == "CT_D"):
80     ct_d.append(lad3)
81     ct_d.append(lad2)
82     ct_d.append(lad1)
83     ct_d.append(lad0)
84
85     if(hex(int("".join(ct_d),2)) == "0x0"):
86         PHASE="ADDR"
```

```
87         else:
88             PHASE="IDLE"
```

**Abbildung 31:** *lpc\_decoder.py* (Zeile 78-88)

Die ADDR-Phase dient dazu, Frames vom TPM zu detektieren. Anders als in den vorherigen Phasen dauert diese Phase vier Takte an. Mit jedem Takt werden vier Bit der 16-Bit Adresse aufgenommen. Wenn mit dem letzten Takt die vollständige Adresse aufgenommen wurde, wird die Zieladresse überprüft. Nur wenn diese der gesuchten Zieladresse entspricht, wird in die nächste Phase gewechselt. Stimmt die Zieladresse nicht mit der erwarteten überein, wechselt das Skript zurück in die IDLE-Phase. Es wird nach TPM-Frames mit den Adressen 0x0024 bis 0x0027 gesucht, weil dort der entsiegelte VMK im Register des TPM abgelegt wird. (Zeile 90-103)

```
90 #ADDR Phase
91 elif(PHASE == "ADDR"):
92     addr.append(lad3)
93     addr.append(lad2)
94     addr.append(lad1)
95     addr.append(lad0)
96
97     if(len(addr) == 16):
98         if(hex(int("".join(addr),2)) in ["0x24",
99 "0x25", "0x26", "0x27"]):
100             PHASE = "TAR"
101         else:
102             PHASE = "IDLE"
103     elif(len(addr) > 16):
104         PHASE = "IDLE"
```

**Abbildung 32:** *lpc\_decoder.py* (Zeile 90-103)

Es folgt die TAR-Phase, die zwei Takte lang geprüft wird. In der TAR-Phase müssen die Signalleitungen LAD0 bis LAD3 auf „high“ gesetzt sein, um die Kontrolle über den Bus vom Host zum TPM zu übergeben. Stellt das Skript fest, dass mindestens eine Signalleitung nicht mehr auf „high“ gehalten wird, wechselt der Zustandsautomat wieder in die IDLE-Phase. Wurde die TAR-Phase zwei Mal erfolgreich durchlaufen, wird in die SYNC-Phase gewechselt. (Zeile 106-119)

```
106 #TAR Phase
107 elif(PHASE == "TAR"):
108     tar.append(lad3)
109     tar.append(lad2)
110     tar.append(lad1)
111     tar.append(lad0)
112
113     if(len(tar) == 8):
114         if(hex(int("".join(tar),2)) == "0xff"):
115             PHASE = "SYNC"
116         else:
117             PHASE = "IDLE"
118     elif(len(tar) > 8):
119         PHASE = "IDLE"
```

**Abbildung 33:** *lpc\_decoder.py* (Zeile 106-119)

Die SYNC-Phase gilt als erfolgreich durchlaufen, wenn alle Signalleitungen für einen Takt auf den Wert „low“ gehalten werden. Wenn das nicht der Fall ist, wird zurück in die IDLE-Phase gewechselt. (Zeile 121-131)

```
121 #SYNC Phase
122 elif(PHASE == "SYNC"):
123     sync.append(lad3)
124     sync.append(lad2)
125     sync.append(lad1)
126     sync.append(lad0)
127
128     if(hex(int("".join(sync),2)) == "0x0"):
129         PHASE = "DATA"
130     else:
131         PHASE == "IDLE"
```

**Abbildung 34:** *lpy\_decoder.py* (Zeile 121-131)

Nach der SYNC-Phase schließt sich die DATA-Phase an. Beim LPC-Frame wird immer ein Byte übertragen. Da pro Takt immer ein Halbbyte übertragen wird, müssen in der DATA-Phase zwei Takte aufgezeichnet werden. Am Ende dieser Phase wird das übertragene Byte in eine Variable gespeichert und der Zustandsautomat in die IDLE-Phase zurückversetzt. Dort wartet das Skript auf den Beginn des nächsten Frames. (Zeile 133-144)

```
133 #DATA Phase
134 elif(PHASE == "DATA"):
```

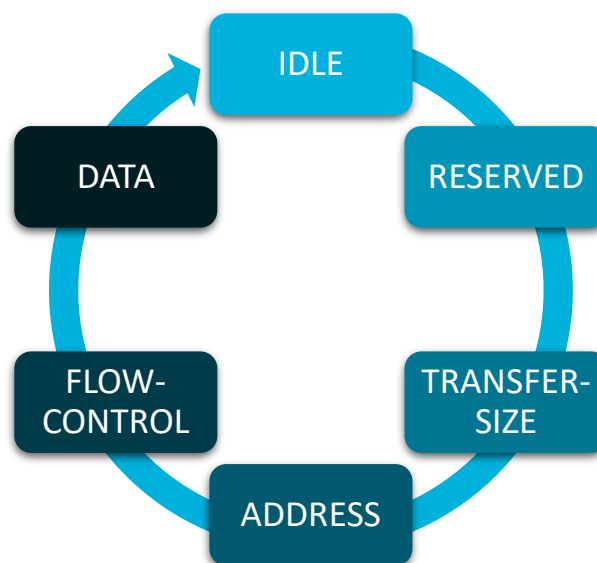
```
135     data.append(lad3)
136     data.append(lad2)
137     data.append(lad1)
138     data.append(lad0)
139
140     if(len(data) == 8):
141         var.stream +=
142         hex(int("".join(data[4:8]),2))[2:]+
143         hex(int("".join(data[0:4]),2))[2:]
144         PHASE = "IDLE"
145     elif(len(data) > 8):
146         PHASE = "IDLE"
```

**Abbildung 35:** *lpc\_decoder.py* (Zeile 133-144)

Wird unabhängig vom aktuellen Zustand des Automaten die LRESET# Signalleitung auf „low“ gezogen, erfolgt immer ein Abbruch. Der Zustandsautomat wird in die IDLE-Phase zurückversetzt.

#### 3.1.5.4 *spi\_decoder.py*

Analog zum „lpc\_decoder.py“ Skript verwendet auch das „spi\_decoder.py“ Skript einen Zustandsautomaten, der die Phasen eines SPI-Frames wie in 2.3.3.2 beschrieben durchläuft. Die Zustände in dem sich der Zustandsautomat befinden kann, sind in **Abbildung 36** dargestellt.



**Abbildung 36:** Zustandsautomat *spi\_decoder.py*



Auch hier wird die Eingabedatei zeilenweise eingelesen und die Werte den entsprechenden Signalleitungen zugeordnet.

Am Anfang befindet sich der Zustandsautomat in der IDLE-Phase und liest auf der steigenden Flanke des Bus-Taktes. Wenn ein Takt anliegt und auf dem ersten Takt der MOSI-Signalleitung der Befehl für einen Lese Zyklus übertragen wird, dann wechselt der Zustandsautomat in die RESERVED-Phase. Vor dem Wechsel werden Variablen aus vorherigen Messungen geleert. (Zeile 64-79)

```
64 #IDLE Phase
65 if(PHASE == "IDLE"):
66     #Prüfe ob Frame eine Lese Transaktion ist
67     if(mosi == "1"):
68         #Beginn ein neues Zyklus, leere alte
Variablen
69         msize.clear()
70         tsize.clear()
71         dsize_tmp = 0
72         asize.clear()
73         fsize.clear()
74         dsize.clear()
75         msize.append(miso) #Zähle Takte im Frame
76         #Lese Phase erkannt, wechsel in die
RESERVED Phase
77         PHASE = "RESERVED"
78     else:
79         PHASE = "IDLE"
```

**Abbildung 37:** spi\_decoder.py (Zeile 64-79)

Die RESERVED-Phase muss durchlaufen werden und führt nach einem Takt unabhängig vom anliegenden Wert auf der MOSI-Signalleitung immer zur TRANSFER-SIZE-Phase (Zeile 81-85)

```
81 #RESERVED Phase
82 elif(PHASE == "RESERVED"):
83     #Bit ist reserviert und kann nicht anders
belegt werden, direkter Wechsel in die nächste Phase
84     msize.append(miso) #Zähle Takte im Frame
85     PHASE = "TRANSFER-SIZE"
```

**Abbildung 38:** spi\_decoder.py (Zeile 81-85)

Die TRANSFER-SIZE-Phase wird für sechs Takte durchlaufen. Mit dem letzten Takt wird in die ADDRESS-Phase gewechselt. Vor dem Wechsel wird ermittelt, wie viele Bytes mit diesem SPI-Frame übertragen werden. Diese Information wird später in der DATA-Phase erneut benötigt, weshalb sie in die Variable „dsize\_tmp“ gesichert wird. (Zeile 87-97)

```
87 #TRANSFER-SIZE Phase
88 elif(PHASE == "TRANSFER-SIZE"):
89     tsize.append(mosi)
90     msize.append(miso)
91
92     if(len(tsize) == 6):
93         #Berechne Anzahl der Bits die übertragen
werden (Später für DATA Phase wichtig!)
94         dsize_tmp = (int("".join(tsize),2)+1)*8
95         PHASE = "ADDRESS"
96     elif(len(tsize) > 6):
97         PHASE = "IDLE"
```

**Abbildung 39:** spi\_decoder.py (Zeile 87-97)

In der ADDRESS-Phase wird die 24-Bit Adresse, an die das SPI-Frame gerichtet ist, aufgezeichnet. Die daraus resultierende Adresse wird mit den entsprechenden Zieladressen eines TPM abgeglichen. Handelt es sich bei der Zieladresse um eine der vier validen Zieladressen, wechselt der Zustandsautomat in die FLOW-CONTROL-Phase. Ist die Adresse an ein nicht TPM-Gerät gerichtet, wird das SPI-Frame verworfen und der Zustandsautomat wechselt zurück in die IDLE-Phase. (Zeile 99-107)

```
99 #ADDRESS PHASE
100 elif(PHASE == "ADDRESS"):
101     asize.append(mosi)
102     msize.append(miso)
103     if(len(aside) == 24):
104         if(hex(int("".join(aside),2)) in
["0xd40024", "0xd40025", "0xd40026", "0xd40027"]):
105             PHASE = "FLOW-CONTROL"
106     elif(len(aside) > 24):
107         PHASE = "IDLE"
```

**Abbildung 40:** spi\_decoder.py (Zeile 99-107)

Für die „FLOW-CONTROL“ Phase ist es entscheidend, dass die Takte auf der MISO-Signalleitung auch mit den vorherigen Phasen bereits aufgezeichnet wurden. Die aufgezeichneten Werte werden in acht Blöcken aufgeteilt und das letzte Bit des letzten Blocks untersucht. Hat dieses den Wert „Eins“, wird in die DATA-Phase gewechselt. Besitzt es den Wert „Null“, dann werden weitere acht Takte aufgezeichnet und der Vorgang wiederholt sich, bis durch das Setzen des letzten Bits auf „Eins“ die Warte Phase beendet wird. (Zeile 109-117)

```
109 #FLOW-CONTROL Phase
110 elif PHASE == "FLOW-CONTROL":
111     #Wenn das letzte Bit eines übertragenen Bytes 1
    ist, dann wechsel in die DATA Phase
112     if(len(msize)%8 == 0 and msize[-1] == "1"):
113         dsize.append(miso)
114         PHASE = "DATA"
115     else:
116         fsize.append(miso)
117         msize.append(miso)
```

**Abbildung 41:** *spi\_decoder.py* (Zeile 109-117)

Die DATA-Phase wird so lange durchlaufen, bis alle zu übertragenden Bits gesendet wurden. Die Information über die Anzahl der zu übertragenden Bits besitzt die Phase aus der TRANSFER-SIZE-Phase. Wurden auf der MISO-Signalleitung alle Bits aufgezeichnet, werden diese in Bytes umgerechnet und in einer Variablen gespeichert. Der Zustandsautomat wechselt am Ende dieser Phase zurück in die IDLE-Phase. (Zeile 119-129)

```
119 #DATA Phase
120 elif PHASE == "DATA":
121     dsize.append(miso)
122     msize.append(miso)
123
124     if(len(dsize) == dsize_tmp):
125         for x in range(0, len(dsize), 4):
126             if(x+4 <= len(dsize)):
127                 var.stream +=
hex(int("".join(dsize[x:x+4]),2))[2:]
128                 PHASE = "IDLE"
129     elif(len(dsize) > dsize_tmp):
130         PHASE = "IDLE"
```

**Abbildung 42:** *spi\_decoder.py* (Zeile 119-129)

Sollten bei der Übertragung Fehler auftreten besteht die Gefahr, dass der Zustandsautomat in einer Phase hängen bleibt und den Beginn eines neuen SPI-Frames verpasst. Aus diesem Grund besitzt das Skript zwei Abbruchbedingungen.

Die erste Abbruchbedingung kontrolliert die Länge des SPI-Frames. Diese wird im Wesentlichen durch die Anzahl der zu übertragenden Bytes und dem Hinzufügen von Warte-Phasen bestimmt. Überschreitet das SPI-Frame die notwendige Größe, dann wird das aktuelle Frame verworfen und in die IDLE-Phase gewechselt.

Die zweite Abbruchbedingung überprüft den Bus-Takt. Beim SPI-Bus setzt dieser aus, wenn kein Frame übertragen wird. Setzt der Bus-Takt über einen zu langen Zeitraum aus, kann dies als Abbruch des Frames gewertet werden. Für die Ermittlung der Zeitspanne, die der Bus-Takt nicht mehr aktiv war, wird die mitgelieferte Zeitspalte der Eingabedatei ausgewertet. Besitzt die Eingabedatei keine Spalte für Aufnahmezeitpunkte, wird anhand der Sample-Rate der zeitliche Abstand zwischen den Messpunkten bestimmt.

## 3.2 Experimentelle Ergebnisse

### 3.2.1 Verwendete Geräte

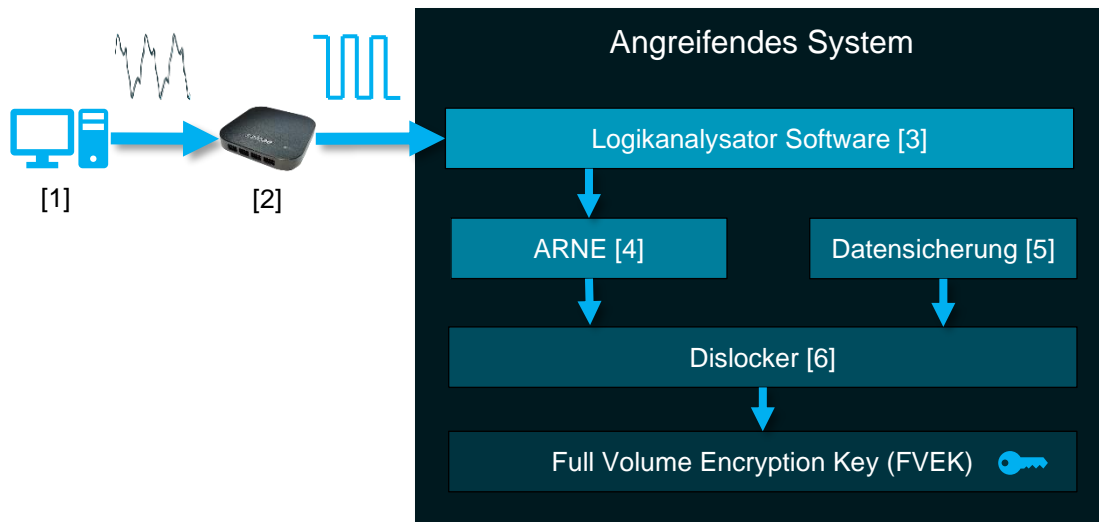


Abbildung 43: Versuchsaufbau

Für die Durchführung des Versuches wird ein Zielsystem (1), ein Logikanalysator (2) und ein angreifendes System mit entsprechender Software (3-6) benötigt. Die Zielsysteme wurden bereits in 3.1.2 vorgestellt. Auf dem angreifenden System befindet sich neben der Logikanalysator Software außerdem der Bus Dekodierer ARNE (4), eine Datensicherung der BitLocker-Partition des Zielsystems (5) und das Programm Dislocker (6). Für die Bereitstellung der notwendigen Software wird ein Linux Betriebssystem auf dem angreifenden System benötigt. In diesem Versuch wird ein angepasstes Arch Linux verwendet.

Als Logikanalysator (2) wird ein Logic Pro 16 der Firma Saleae gewählt. Dieser verfügt über 16 Kanäleingänge und kann damit sowohl die Signalleitung des LPC-, sowie die des SPI-Buses einlesen. Die aufgenommenen Daten des Logikanalysators werden direkt über USB an das angeschlossene Computersystem gesendet. Die maximale Aufnahmelänge wird nur über die Größe des Arbeitsspeichers des angeschlossenen Systems begrenzt. Die Abtastrate richtet sich nach der Anzahl der Signaleingänge und der verwendeten

Logikanalysator Software (3). Mit dem mitgelieferten Saleae Logic 2 Programm beträgt die Abtastrate beim LPC-Bus maximal 250 MS/s und beim SPI-Bus 500 MS/s. Neben der zum Logikanalysator mitgelieferten Software soll in diesem Versuch auch erprobt werden, ob der Angriff mit freier Software durchführbar ist. Hierzu wird das Programm PulseView verwendet, das eine Unterstützung für zahlreiche andere Logikanalysatoren bietet. Die Logikanalysator Software muss in der Lage sein, die aufgenommenen Daten in Form einer Comma-separated values (CSV) Datei exportieren zu können, damit diese an den Bus-Dekodierer übergeben werden kann.

Das Programm ARNE (4) benötigt eine funktionierende Python 3 Umgebung mit entsprechend installierten Python Bibliotheken. Es erhält als Eingabedatei die exportierte CSV-Datei der Logikanalysator Software, dekodiert den aufgenommenen Stream und extrahiert aus diesem den entsiegelten VMK.

Für die Extraktion des FVEK und die Verifikation des entsiegelten VMK aus dem Programm ARNE wird die BitLocker-Partition benötigt. Für die Reproduzierbarkeit der Ergebnisse dieser Arbeit wird vor dem Angriff jeweils eine Datensicherung der Partition mit dem Programm FTK Imager erstellt. FTK Imager befindet sich hierbei in der Version 4.7.1.2.

Um aus dem entsiegelten VMK den dazugehörigen FVEK extrahieren zu können, wird das Programm Dislocker (6) ab der Version 0.7.2 benötigt. Dislocker wird ursprünglich verwendet, um auf Linux und Mac OSX Betriebssystemen das Einbinden von BitLocker-Partitionen zu ermöglichen. Für diesen Versuch wird die Funktion zum Extrahieren des FVEK benötigt. Dazu muss der entsiegelte VMK und eine Datensicherung der BitLocker-Partition dem Programm übergeben werden.

Eine fertig eingerichtete Linux Arch Distribution mit allen notwendigen Paketen und Programmen ist dieser Arbeit beigelegt.

### 3.2.2 Vorgehen

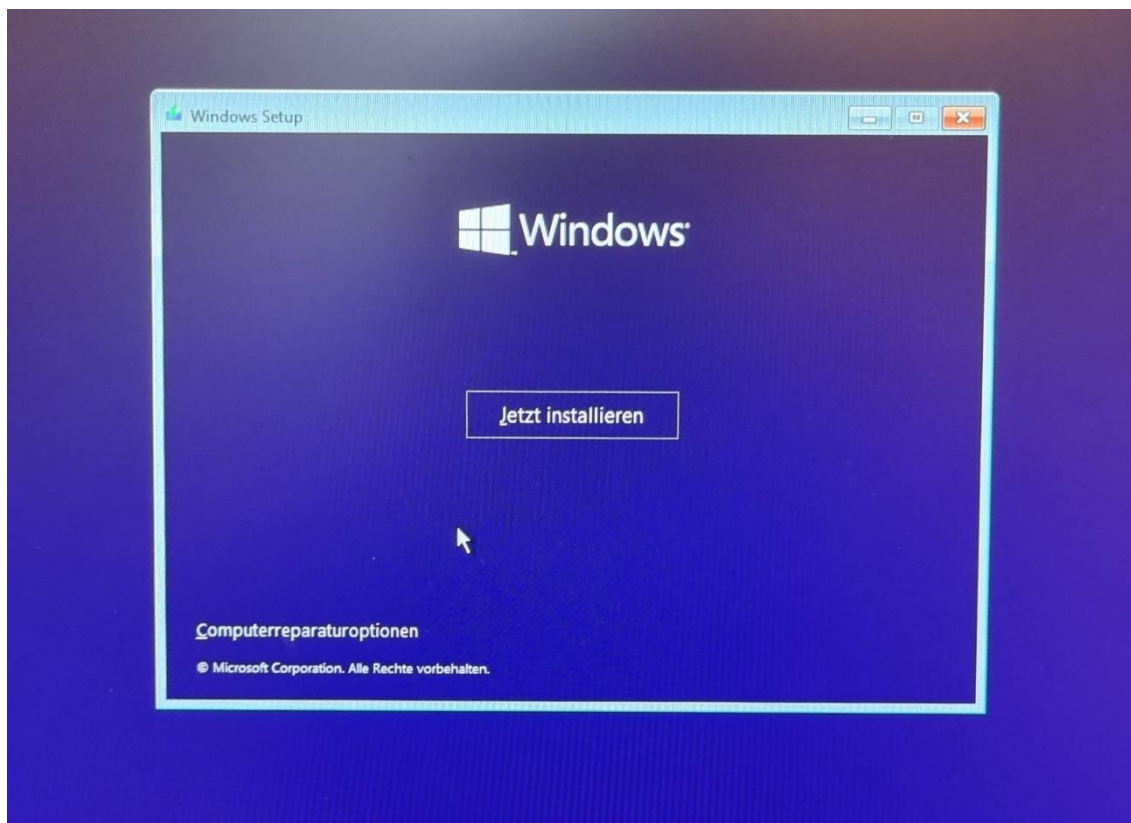
Das folgende Vorgehen beschreibt den Ablauf einer Messung. Für den Versuch werden mehrere Messungen mit verschiedenen Betriebssystem- und Zielsystem-Kombinationen durchgeführt. Die Ergebnisse der Messungen werden anschließend im Abschnitt 3.2.3 analysiert und ausgewertet.

Auf dem Zielsystem wird im ersten Schritt das entsprechende Betriebssystem installiert. In der **Abbildung 44** ist beispielhaft die Installation eines Windows 8.1 dargestellt. Bei der Installation werden alle Einstellungen im Standard belassen. Als Benutzerkonto wird immer ein lokales Konto mit folgenden Benutzerdaten gewählt:

Benutzer: tpm

Passwort: qwertz

Nach der Installation des Grundsystems werden alle verfügbaren Updates in das System eingespielt, welche die Sicherheit des Betriebssystems erhöhen. Bei den von Microsoft noch unterstützten Betriebssystem (>= Windows 8.1) können die Updates, über den Microsoft Update Server, automatisch vom Betriebssystem bezogen werden. Bei den älteren Betriebssystemen, wie Windows Vista, besteht keine Verbindung mehr zum Microsoft Update Server. Hier müssen die Updates zuvor manuell heruntergeladen und anschließend offline eingespielt werden. Befindet sich das System auf einem aktuellen Stand, wird mit der Einrichtung von BitLocker fortgefahren. Auch hier werden alle Windows Einstellungen im Standard belassen, was dazu führt, dass das Betriebssystem als Schutzvariante für BitLocker das TPM wählt. Wenn möglich, wird immer nur der verwendete Speicherplatz mit BitLocker verschlüsselt.



**Abbildung 44:** Windows 8.1 Installation

Nach der Einrichtung des Betriebssystems wird mit dem Befehl `manage-bde.exe` Informationen über BitLocker gesammelt, um zu ermitteln, welche versionsspezifische Veränderungen es mit den unterschiedlichen Betriebssystemen gegeben hat. Der Befehl, für den Admin-Rechte notwendig sind, wird über die Kommandozeile des Windows Betriebssystems abgesetzt. Mit dem in **Abbildung 45** dargestellten Befehl kann ermittelt werden, in welcher Version sich das BitLocker Konfigurationstool befindet, welche BitLocker Version verwendet wird und welcher Verschlüsselungsalgorithmus von BitLocker im Standard gesetzt ist. Der in **Abbildung 46** gezeigte Befehl liefert Rückmeldung über die Schutzvorrichtung der BitLocker-Partition. Dabei werden auch aufgeführt, welches PCRs BitLocker für die Validierung verwendet.



```
1 manage-bde.exe -status
2 BitLocker-Laufwerkverschlüsselung:
3 Konfigurationstool, Version 6.3.9600
4 Copyright (C) 2013 Microsoft Corporation. Alle
5 Rechte vorbehalten.
6
7 Volume "C:" []
8 [Betriebssystemvolumen]
9
10      Gerät:                238,13 GB
11      BitLocker-Version:    2.0
12      Konvertierungsstatus: Nur verwendeter
13 Speicherplatz ist verschlüsselt
14      Verschlüsselt (Prozent): 100,0 %
15      Verschlüsselungsmethode: AES 128
16      Schutzstatus:        Der Schutz ist
17      aktiviert.
18      Sperrungsstatus:     Entsperrt
19      ID-Feld:             Unbekannt
20      Schlüsselschutzvorrichtungen:
21          TPM
22          Numerisches Kennwort
```

**Abbildung 45:** `manage-bde.exe -status` (Esprimo Windows 8.1)

```
1 Manage-bde.exe -protectors -get C:
2 BitLocker-Laufwerkverschlüsselung:
3 Konfigurationstool, Version 6.3.9600
4 Copyright (C) 2013 Microsoft Corporation. Alle
5 Rechte vorbehalten.
6
7 Volume "C:" []
8 Alle Schlüsselschutzvorrichtungen
9
10      TPM:
11      ID: {8608B056-CE11-47B9-961A-85A218ED84B6}
12      PCR-Validierungsprofil:
13      0, 2, 4, 8, 9, 10, 11
14
15      Numerisches Kennwort:
16      ID: {D8968DCA-E992-4850-9EFD-1405BD4F3FDB}
17      Kennwort:
18      125026-516560-431332-133639-194007-249381-
19      372922-252714
```

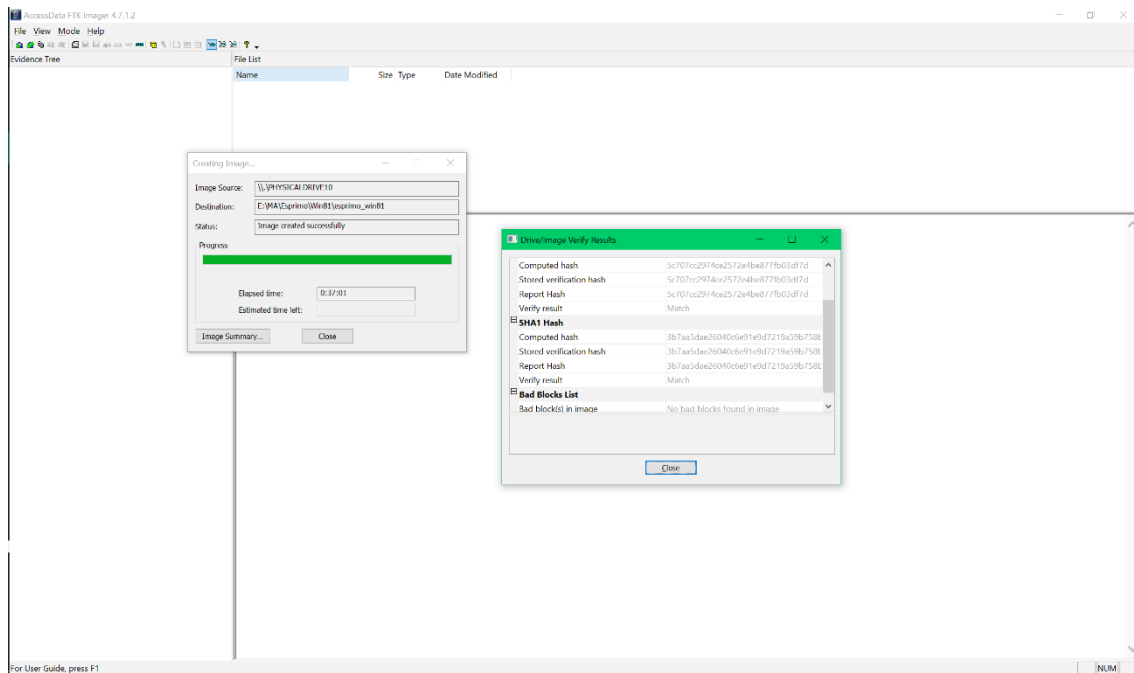
**Abbildung 46:** `manage-bde.exe -protectors -get C:` (Esprimo Windows 8.1)

Für den Angriff wird eine Datensicherung des gesamten Datenträgers erstellt. Um den Einfluss des Betriebssystems und BitLockers auf die Größe der Datensicherung bestimmen zu können, muss dafür gesorgt werden, dass die Daten im gelöschten Bereich aus vorherigen Installationen nicht mit gesichert werden. Um nicht genutzte Dateien aus dem freien Speicherbereich des Datenträgers zu entfernen, wird das Programm `sdelete64` verwendet. Bei diesem handelt es sich um ein erweitertes Systemprogramm aus den Sysinternals-Tools von Mark Russinovich. [61] Der Befehl, der in **Abbildung 47** dargestellt ist, ermöglicht mit dem Parameter `/z` ein Überschreiben des nicht genutzten Speicherbereiches der Betriebssystemplatte. `sdelete64` überschreibt den freien Bereich mit Nullen.

```
1 C:\Users\tpm> sdelete64.exe /z C:  
2 Copyright (C) 1999-2019 Mark Russinovich  
3 Sysinternals - www.sysinternals.com  
4  
5 SDelete is set for 1 pass.  
6 Cleaning free space on C:\: 2%
```

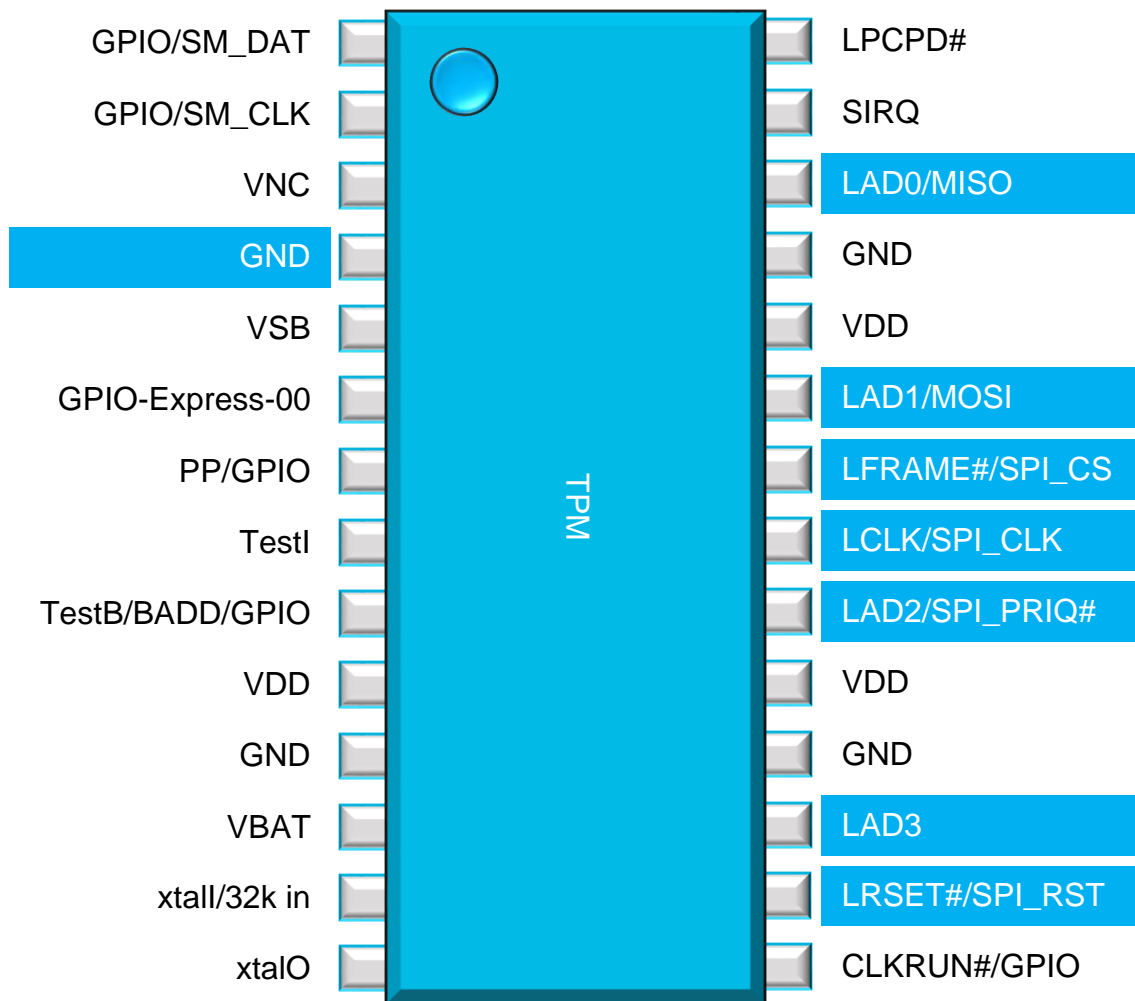
**Abbildung 47:** Ausgabe `sdelete64.exe`

Nachdem der Datenträger vorbereitet ist, wird mit der Datensicherung fortgefahren. Dazu muss der Datenträger aus dem System entnommen und über einen Hardware-Schreibschutz mit dem Auswertesystem verbunden werden. An diesem wird mit Hilfe der Software FTK Imager von Access Data eine Datensicherung im Expert Witness Disk Image Format (EWF) erzeugt. Dieses Format wird verwendet, um Datenträgerabbilder für forensische Zwecke komprimiert zu sichern. Kann, wie im Fall des Microsoft Surface Pro 5, kein Datenträger entnommen werden, da dieser verlötet ist, erfolgt die Datensicherung auf einem anderen Weg. Hier muss das Zielsystem über ein Fremdbetriebssystem gebootet werden. Dazu eignet sich beispielsweise ein angepasstes Windows PE, auf welchem ebenfalls die Software FTK Imager installiert ist. Nachdem das Windows PE gebootet ist, wird ein zusätzlicher externer Datenträger angeschlossen, auf dem die Datensicherung gespeichert werden kann.



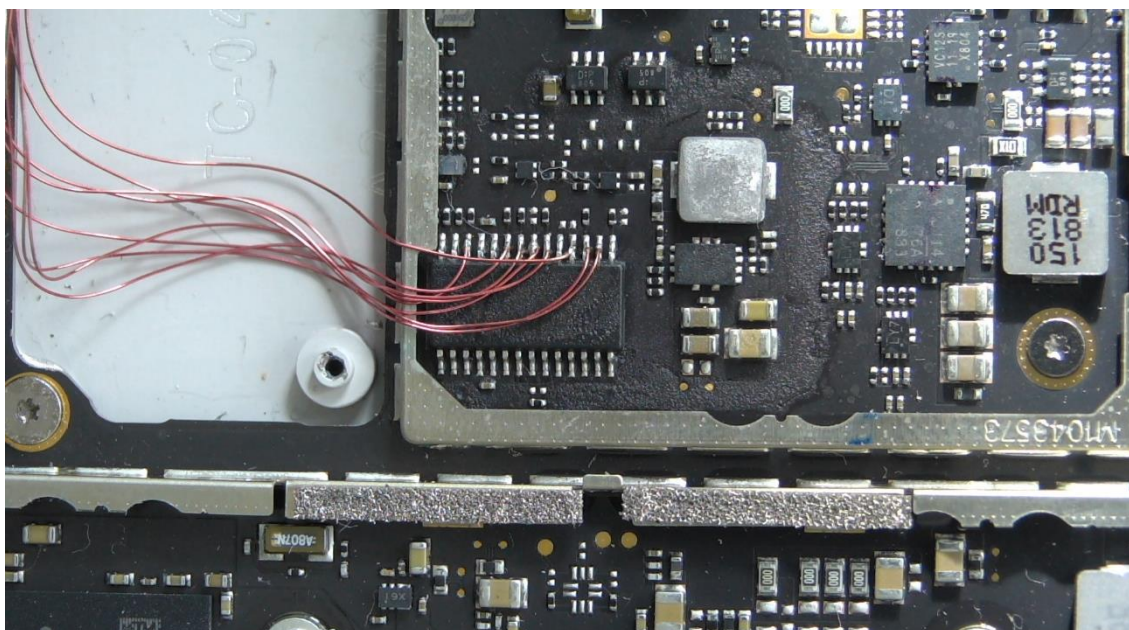
**Abbildung 48:** FTK Imager

Um die Kommunikation am Bus mithören zu können, ist es erforderlich, die benötigten Signalleitungen zu kontaktieren. Dazu muss zunächst die Pinbelegung des entsprechenden TPM ermittelt werden. Da es, wie im Abschnitt 2.1.4 beschrieben, von der TCG Empfehlungen zur Pinbelegung gibt, kann sie in den meisten Fällen auch auf das vorhandene TPM übertragen werden. In der **Abbildung 49** ist das vorgegebene Pinout der TCG dargestellt. Die benötigten Signale sind farblich hervorgehoben. Für den LPC-Bus werden die Signalleitungen LAD0, LAD1, LAD2, LAD3, LFRAME#, LCLK und optional LRESET# benötigt. Kommuniziert das TPM über einen SPI-Bus, genügen die Signalleitungen MISO, MOSI, SPI\_CS und SPI\_CLK. Das Masse-Signal (GND) kann neben dem Pin 4 auch alternativ an den Pins 11, 18 oder 25 abgegriffen werden.



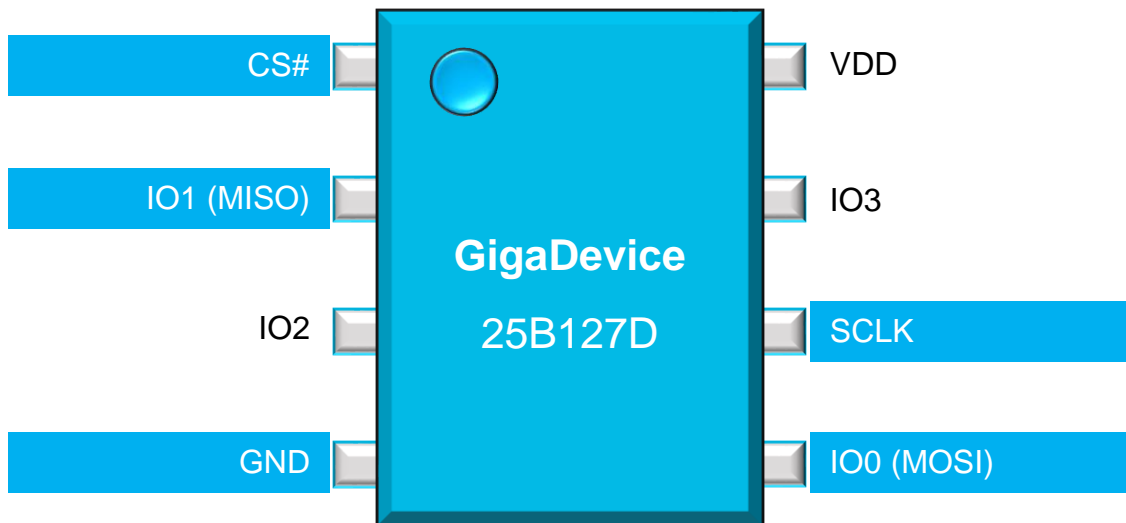
**Abbildung 49:** TPM Pinout nach TCG-Spezifikation (angelehnt an [5])

Zum Abgreifen der Signalleitungen werden dünne Kupferlackdrähte mit einem Durchmesser von 0,2 mm an den entsprechenden Pins des TPMs angelötet. In der **Abbildung 50** sind solche Drähte an das TPM des Microsoft Surface 5 Pro angelötet. Die Enden der Drähte sind zu einer Buchsenleiste am Gehäuserand des Gerätes geführt, um dort die Signale mit einem Logikanalysator abgreifen zu können.

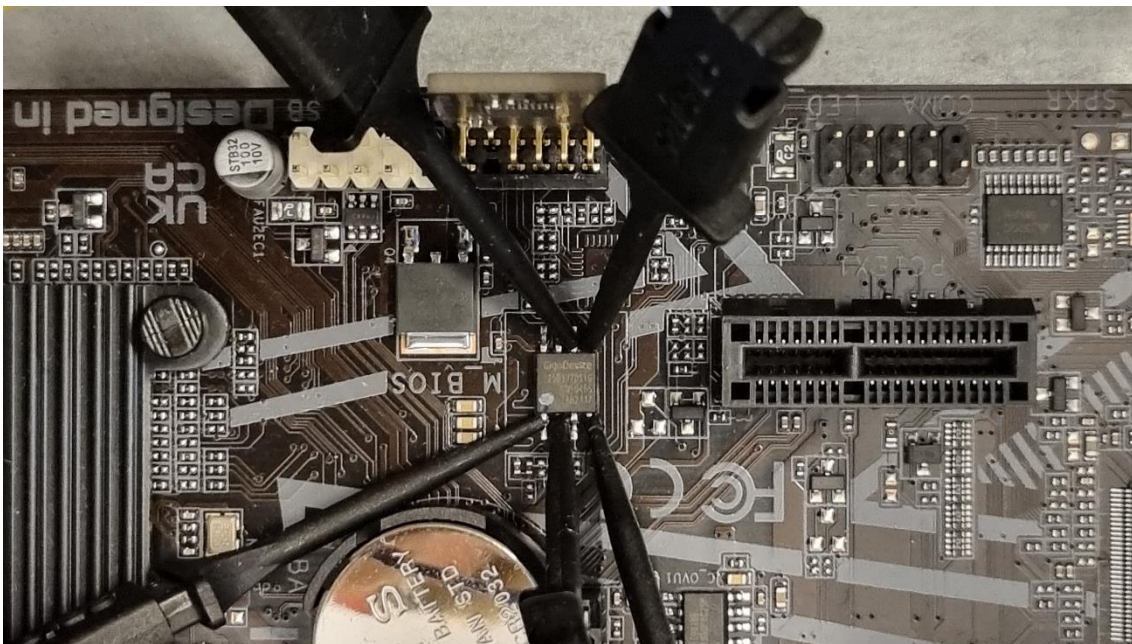


**Abbildung 50:** Kontaktierung des Microsoft Surface Pro 5

Statt mechanisch durch Anlöten auf das TPM einzuwirken, können alternativ die Signale auch durch entsprechende Klemmen an einem anderen Bus Teilnehmer abgegriffen werden. Am Beispiel des Gigabyte H410M S2H V3 Zielsystems soll untersucht werden, ob das Mitlesen am Firmware-Chip ebenfalls zu einem erfolgreichen Abhören der Kommunikation zwischen System und TPM führt. Dabei wird die Eigenschaft eines Bus-Netzwerkes genutzt, dass sich alle Teilnehmer die selben Signalleitungen teilen. Für die Kontaktierung muss aus dem Datenblatt des Firmware-Chips (GigaDevice 25B127DSIG) die Pins zum SPI-Bus ermittelt und mit den Klemmen kontaktiert. Die **Abbildung 51** zeigt das Pinout des Firmware-Chips. Es werden die Pins 1, 2, 4, 5 und 6 benötigt. In der **Abbildung 52** sind die angebrachten Klemmen am Firmware-Chip dargestellt.



**Abbildung 51:** Pinout GigaDevice 25B127D [62]



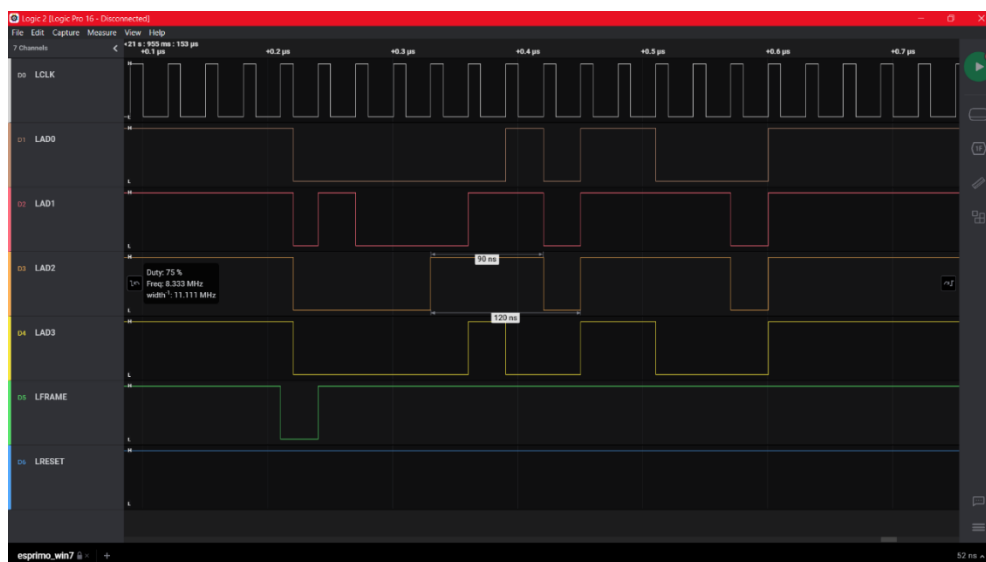
**Abbildung 52:** Kontaktierung Gigabyte H410M S2H V3 mit Klemmen am Firmware-Chip

Das Arbeiten am geöffneten Computersystem erfordert entsprechende Sicherheitsmaßnahmen, da Kurzschluss und im Bereich der Spannungsversorgung Lebensgefahr besteht. Beim Anlöten bzw. Anklemmen der Drähte an den jeweiligen Bus muss darauf geachtet werden, dass dies nur im ausgeschalteten Zustand erfolgen darf. Zusätzlich müssen Arbeitsgeräte und



ausführende Personen entsprechend geerdet sein, um eine Beschädigung der Halbleiter zu vermeiden. Während des Versuches muss darauf geachtet werden, dass der Versuchsaufbau nicht berührt oder bewegt werden darf. Es sollte ein entsprechendes Hinweisschild am Versuchsaufbau angebracht sein.

Die Signalleitungen werden zum Saleae Logic Pro 16 Logikanalysator geführt, der die Signale misst, aufbereitet und über USB3.0 an das Auswerte System sendet. Die Einstellungen und Auswertung des Logikanalysators erfolgt über entsprechende Software am Auswerterechner. Die Messung beginnt mit dem Einschalten des Zielsystems und wird nach dem erfolgreichen Startvorgang des Windows Betriebssystems beendet. Dabei erfolgen die Messungen mit den Programmen Saleae Logic 2 (**Abbildung 53**) und PulseView (**Abbildung 54**). Bei dem Saleae Logic 2 werden als Abtastrate 125 MS/s und eine Schwellspannung 3,3V für „low“ und „high“ gewählt. Bei PulseView wird die maximale Abtastrate von 50 MS/s eingestellt. Das Einstellen einer Schwellspannung ist unter PulseView nicht möglich.



**Abbildung 53:** Saleae Logic 2



**Abbildung 54:** PulseView

Die erfolgreiche Messung kann in der Logikanalysator Software eingesehen werden. Sowohl PulseView als auch Saleae Logic 2 boten die Möglichkeit, die aufgenommene Messung in eine Comma-Separated-Values (CSV) Datei zu exportieren. In der **Abbildung 55** ist der Inhalt der Messung am Gigabyte H410M S2H V3 Zielsystem mit einem Windows 11 Betriebssystem auszugsweise dargestellt. Die erste Zeile der Datei definiert die Überschriften der einzelnen Spalten. Die Datei besitzt fünf Spalten, wobei die Spalte eins die Zeitspalte darstellt und die Spalten zwei bis fünf die Signalleitungen des SPI-Buses. Zu jedem Zeitpunkt besitzt eine Signalleitungen einen eindeutigen „high“ (1) oder „low“ (0)-Zustand.

1	Time [s], SCLK, CS, MOSI, MISO
2	4.007206890, 1, 0, 1, 1
3	4.007206900, 0, 0, 1, 1
4	4.007206930, 1, 0, 1, 1
5	4.007206940, 0, 0, 0, 1
6	4.007206960, 0, 0, 1, 1
7	4.007206970, 1, 0, 1, 1
8	4.007206980, 0, 0, 0, 1
9	4.007207000, 0, 0, 1, 1
10	4.007207030, 1, 0, 1, 1
11	4.007207040, 0, 0, 0, 0
	...

**Abbildung 55:** Inhalt Gigabyte Win 11 HLA CSV



Die exportierte CSV-Datei wird mit dem erstellten Programm ARNE eingelesen. Dabei wird der Bus dekodiert und nach der Signatur eines entsiegelten VMK durchsucht. In der **Abbildung 56** wird die Messung des Surface Pro 5 mit einem Windows 10 Betriebssystem, die mit PulseView aufbereitet wurde, eingelesen. Wird innerhalb des aufgezeichneten Streams der entsiegelte VMK gefunden (Zeile 15-16), so wird dieser in eine externe Datei gespeichert.

```
1 python main.py -i
  D:\Nextcloud\Uni\Masterarbeit\MA\Versuch\Angriff\Surface\Win10\Surface_Win10_50MHz_pulseview.csv -k
  "1=LCLK,2=LAD0,3=LAD1,4=LAD2,5=LAD3,6=LFRAME,7=LRESET" -nt 50000000 -o Surface_Win10.vmk
2 +-----+
3 |
4 |   ARNE
5 |
6 |
7 |
8 |
9 |
10 | (c) Sebastian Lasogga
11 +-----+
12
13 + Ermittle Messpunkte...444951969
14 + Starte LPC-Decoder...
15 + VMK Signatur gefunden: 2c000000100000003200000
16 + Entsiegelten VMK gefunden:
  55f0fd7e658f1d8aed7ce2e0339cb7e86f44bbcd55d012d5e9
  a9483df07624
17 + Beende LPC-Decoder
18 + Fortschritt: 42 %, TPM Frames: 1819
19 + Es konnte ein TPM Stream ermittelt werden...
```

**Abbildung 56:** ARNE

Um aus dem entsiegelten VMK den FVEK extrahieren zu können, wird nun die angefertigte Datensicherung benötigt. Da diese im EWF-Dateiformat vorliegt, muss diese zunächst in ein Rohdatenformat (RAW) umgewandelt werden. Auf dem Auswertesystem erfolgt dies mit dem `xmount` Befehl und ist in **Abbildung 57** dargestellt. Dem Programm wird das Einleseformat (EWF) und die Datei übergeben. Um eine Veränderung an der Datensicherung zu unterbinden, wird eine Cache Datei erzeugt, in die stattdessen alle Änderungen geschrieben

werden. Zum Schluss erfolgt die Angabe, des Ausgabeformats der Datensicherung, sowie der Zielort der konvertierten Datei.

```
1 sudo xmount --in ewf Surface_win10.E0? --cache  
/tmp.xml --out raw /tmp/xm/
```

**Abbildung 57: XMOUNT**

Die Partitionen des Images im RAW-Format werden anschließend in das Betriebssystem mit dem Befehl `losetup` eingebunden, um mit Dislocker direkten Zugriff auf die BitLocker-Partition zu erhalten. Der Befehl `losetup` ist in **Abbildung 58** dargestellt und kann mit dem Parameter `--partscan` selbstständig nach Anfang und Ende der einzelnen Partitionen suchen. Der Zusatz `--find` sorgt dafür, dass das Programm automatisch nach einem freien Loop-Device sucht, in dem es die Partitionen einbinden kann. Der Parameter `--show` gibt den Wert des Loop-Devices zurück, in dem die Partitionen eingebunden wurden.

```
1 sudo losetup --partscan --find --show  
/tmp/xm/Surface_win10.dd  
2 /dev/loop15
```

**Abbildung 58: losetup**

Im letzten Schritt des Versuches wird dem Programm `Dislocker` der Pfad zur eingebundenen BitLocker-Partition und dem entsiegelten VMK mitgeteilt. Im Standard öffnet `Dislocker` die Partition ohne weitere Ausgabe. Mit dem Parameter `-vvvv` kann eine detaillierte Ausgabe von `Dislocker` erzwungen werden, die bei erfolgreichem Öffnen der BitLocker-Partition auch den extrahierten FVEK ausgibt. (Zeile 10-13)

```

1  sudo dislocker -vvvv -V /dev/loop15p3 -K
   surface_win10.vmk /tmp/bitlocker/
2  Tue Aug 23 16:56:47 2022 [DEBUG] Verbosity level to
   DEBUG (4) into 'stdout'
3  Tue Aug 23 16:56:47 2022 [INFO] dislocker by Romain
   Coltel, v0.7.2 (compiled for Linux/x86_64)
4  Tue Aug 23 16:56:47 2022 [INFO] Compiled version:
   master:44ea0da
5  Tue Aug 23 16:56:47 2022 [DEBUG] --- Config...
6  Tue Aug 23 16:56:47 2022 [DEBUG]     Verbosity: 4
7  Tue Aug 23 16:56:47 2022 [DEBUG]     Trying to
   decrypt '/dev/loop15p3'
8  Tue Aug 23 16:56:47 2022 [DEBUG]           using the VMK
   file at 'surface_win10.vmk'
9  ...
10 Tue Aug 23 16:56:47 2022 [DEBUG] FVEK -----
   -----
11 Tue Aug 23 16:56:47 2022 [DEBUG] 0x00000000 48 85 5b
   fd ab e4 69 78-bb 10 ea 18 e4 9c 43 c0
12 Tue Aug 23 16:56:47 2022 [DEBUG] 0x00000010 7f 6a 9f
   29 f4 10 00 e0-c1 05 7b 18 de 35 d3 43
13 Tue Aug 23 16:56:47 2022 [DEBUG] -----
   -----

```

**Abbildung 59:** Dislocker

In der **Tabelle 18** sind die veränderlichen Parameter zu diesem Versuch aufgeführt. Insgesamt werden 30 Messungen vorgenommen, bei den das Zielsystem, das Betriebssystem und die Logikanalysator Software verändert werden.

Parameter	Messwerte
Zielsystem	[ Fujitsu Siemens Esprimo E5720, Microsoft Surface Pro 5, Gigabyte H410M SH2 V3 ]
Betriebssystem	[ Vista, Win7, Win8, Win8.1, Win10, Win11 ]
Secure Boot	[ Ein, Aus ]
Software	[ Saleae Logic 2, PulseView]
Gesamt	30

**Tabelle 18:** Veränderliche Parameter

### 3.2.3 Auswertung

Der Versuch wurde mit entsprechend angepassten Parametern mehrmals wiederholt. In der Tabelle 19 sind die erfolgreich extrahierten entsiegelten VMKs und FVEKs dargestellt. In der ersten Spalte der Tabelle sind die Parameter des Versuchs aufgeführt. Auf die gewonnenen Ergebnisse und gefundenen Randbedingungen wird nachfolgend eingegangen.

Als eine Randbedingung für diesen Versuch wurde der Logikanalysator und die dazugehörige Software festgelegt. Bei der Wahl der Logikanalysator Software konnten Unterschiede festgestellt werden. Der Angriff war beim Fujitsu Siemens Esprimo E5720 Zielsystem nicht erfolgreich, wenn als Software PulseView gewählt wurde. Ursächlich dafür ist die begrenzte Abtastrate von PulseView. Wie in 3.1.4.1 gezeigt wurde, beträgt die Taktfrequenz des Fujitsu Siemens Esprimo E5720 33 MHz. Mit der maximalen Abtastrate von 50 MS/s, die PulseView in Kombination mit dem Saleae Logic Pro 16 bereitstellt, wird das Abtasttheorem verletzt und der Alias-Effekt trifft ein. Ein Dekodieren der LPC-Frames ist in diesem Fall nicht möglich.

Parameter	Signatur	VMK	FVEK
<b>AES 128 mit Diffuser</b>			
Esprimo	2c 00 00 00 01 00 00 00	aa c3 ca b1 f6 75 cf a0	ed 5d 8c 13 34 9c 7d b8
Vista	03 20 00 00	61 43 91 e5 74 86 13 fe	c2 04 e6 2e 25 b5 98 f1
Saleae Logic 2		b4 ae 72 f2 f1 4b 95 9b 15 b1 41 88 e1 d1 1b 0a	cc cc 97 4f 1b d5 20 e6 f2 07 07 58 f3 ce e9 60 b7 1b 0f 8d dc 5d fc 2a 57 49 e1 d6 d9 42 c7 38 95 fb 49 1c 4c ae 06 3f d7 fe 98 29 e0 b4 ad 8c
Esprimo	2c 00 00 00 01 00 00 00	5d 9b 73 a8 00 04 b2 fe	d5 f3 21 61 36 4f 0c 3f
Win7	03 20 00 00	55 83 f1 9e 23 7c a7 63	39 7f 0c 0d c0 94 c7 03
Saleae Logic 2		85 69 51 97 25 c3 31 8c fc 4b 03 60 1b 4f c7 48	5b a1 ce 6c 8c ad 24 4d 78 33 66 f4 24 64 2a 91 c6 c7 41 d0 63 4e 41 97 46 a2 aa 5b ee 77 c0 99 7c 7d d1 4d 2a 9d 4e d1 00 a8 4b d8 38 7f 45 32
<b>AES 128</b>			

Esprimo	2c 00 00 00 01 00 00 00	ea 4f 98 76 2a 23 08 04	d8 0b 0f 3d ae 36 64 2f
Win8.1	03 20 00 00	00 a0 48 34 33 ac 1f 00	d2 4a 59 45 64 2e f2 93
Saleae Logic 2		3d 2a ce 4d b2 de 29 ea	
PulseView		79 53 48 61 3f ca 5e 63	
Gigabyte	2c 00 00 00 01 00 00 00	98 af 7e 0a 75 c1 64 89	53 1f 6a a7 e0 41 cf 87-
Win8.1	03 20 00 00	c7 2b 05 fd 50 92 76 8f	ba 82 d1 dd c1 88 e7 fd
Saleae Logic 2		a8 2b 6d 91 83 5d 46 92	
PulseView		49 d1 cd 8e 88 78 c4 07	

## XTS-AES 128

Esprimo	2c 00 00 00 01 00 00 00	8b a7 65 5e 63 af 86 6c	7a cd d3 27 96 12 d1 b2
Win10	03 20 00 00	49 e4 46 c4 e3 6e c9 f5	50 90 3d e4 56 85 82 eb
Saleae Logic 2		08 0d 6d 87 19 05 b9 a8	ed e3 b9 a8 27 09 ea a1-
PulseView		6a 02 74 36 a2 d2 41 66	38 7e 85 11 b3 3c 14 a0
Gigabyte	2c 00 00 00 01 00 00 00	2d ce 7c f6 47 ba 20 f6	6c 2f 33 ea 3b 41 9c c8-
Win10	03 20 00 00	74 d0 54 39 83 74 64 ff	95 09 d8 54 fa 68 f5 b1
Saleae Logic 2		b8 b5 80 f8 60 04 df 9b	62 6e 77 27 d8 22 4d 60-
PulseView		a7 e9 87 68 ba a6 0b 95	70 42 12 9f 14 8f 1e d2
Surface	2c 00 00 00 01 00 00 00	55 f0 fd 7e 65 8f 1d 8a	48 85 5b fd ab e4 69 78
Win10	03 20 00 00	ed 7c e2 e0 33 9c be b7	bb 10 ea 18 e4 9c 43 c0
Secure Boot			

Saleae Logic 2 PulseView		e8 6f 44 bb cd 55 d0 12 d5 e9 a9 48 3d f0 76 24	7f 6a 9f 29 f4 10 00 e0 c1 05 7b 18 de 35 d3 43
Gigabyte Win11 Saleae Logic 2 PulseView	2c 00 00 00 01 00 00 00 03 20 00 00	85 a4 db 72 be 66 99 24 12 b0 eb 22 ce fa bf 25 c6 d9 f7 6b a9 af 3a bf 50 d2 99 d9 04 50 17 ea	aa 89 85 d0 3e eb fb f6- 82 2e 44 1f e9 65 0f 1f 78 8c b5 c3 c6 2b 30 8b- 5d 75 c5 d0 00 eb 40 d1
Surface Win11 Secure Boot Saleae Logic 2 PulseView	2c 00 00 00 01 00 00 00 03 20 00 00	33 10 19 c2 8b d7 52 eb a3 41 29 e0 be 28 f5 c4 c2 bb 52 1e a2 4e 44 66 64 9b a3 e7 b8 1e c4 e9	6b 40 37 1b ce 35 78 77 8e 80 88 58 ed 8c 35 e9 78 cc 6e 00 d6 b4 74 f4- b5 95 ef 49 94 c4 9f 9a

**Tabelle 19:** Einfluss Verschlüsselungsalgorithmus auf entsiegelten VMK

Bei dem Versuch wurde weiterhin festgestellt, dass es in den Windows Betriebssystemen Veränderungen an BitLocker gegeben hat. Die Veränderungen wurden anhand der aufgezeichneten Ausgaben des BitLocker-Konfigurationstools untersucht. Sie sind in **Tabelle 20** dargestellt. Windows Vista liefert bis auf den Verschlüsselungsalgorithmus keine Informationen zu BitLocker. Seit Windows 8 befindet sich BitLocker in der Version 2.0 und wurde nicht mehr verändert. Versionsänderungen gab es nur noch am BitLocker Konfigurationstool selbst. Hinsichtlich seines Standard-Verschlüsselungsalgorithmus wurde BitLocker mehrfach verändert. Unter Windows Vista und Windows 7 ist die AES-Verschlüsselung mit Diffuser als Standard festgelegt. Mit der Einführung von Windows 8 wurde die AES-Verschlüsselungsmethode mit Diffuser abgeschafft, sodass der Benutzer im Standard nur noch eine 128-Bit AES Verschlüsselung erhält. Ab Windows 10 wurde der Block Cipher Mode XTS mit AES als Standard eingeführt.

Betriebssystem	BitLocker Version	BitLocker Konfigurationstool	Verschlüsselungstyp
Vista	k.A	k.A.	AES 128 mit Diffuser
Win7	Windows 7	6.1.7601	AES 128 mit Diffuser
Win8	2.0	6.2.9200	AES 128
Win8.1	2.0	6.3.9600	AES 128
Win10	2.0	10.0.19041	XTS-AES 128
Win11	2.0	10.0.22000	XTS-AES 128

**Tabelle 20:** Einfluss Betriebssystem auf BitLocker

Um einen Einfluss des Verschlüsselungsalgorithmus auf den Versuchsaufbau zu untersuchen, wurden die gewonnenen Schlüssel in **Tabelle 19** miteinander verglichen. Am entsiegelten VMK konnten keine Veränderungen durch den Verschlüsselungsalgorithmus festgestellt werden. Dieser verfügt immer über die feste Länge von 44 Bytes mit einer 12 Byte langen Signatur. Die Signatur selbst behält dabei ihre Struktur immer bei. Das neunte Byte in der Signatur definiert



den Verschlüsselungstyp. Dieser ist erwartungsgemäß bei allen aufgenommenen entsiegelten VMK mit 0x03 definiert, was auf eine Verschlüsselung mit TPM hindeutet. Hinsichtlich des Schlüsselteils im entsiegelten VMK hat dieser ebenfalls immer die Länge von 32 Bytes. Da sich die Signatur des entsiegelten VMK nicht verändert hat, war es mit dem Programm ARNE unabhängig vom Betriebssystem immer möglich diesen zu extrahieren. Beim FVEK konnten hingegen Unterschiede aufgrund des Verschlüsselungsalgorithmus festgestellt werden. Bei einer AES 128-Bit Verschlüsselung mit Diffuser, welche bei Windows Vista und Windows 7 zum Einsatz kommt, besitzt der FVEK eine Gesamtlänge von 512-Bit (64 Byte). Dieser besteht aus einem 128-Bit langen AES-Schlüssel und einem 384-Bit langen Schlüssel für den Diffuser Algorithmus. Unter Windows 8 und 8.1 wird eine normale 128-Bit AES-Verschlüsselung verwendet. Hier besitzt der FVEK erwartungsgemäß die Länge von 128-Bit (16 Byte). Ab Windows 10 besteht der FVEK aus einem 128-Bit AES Schlüssel und aus einem 128-Bit langen XTS-Schlüssel. Dadurch ergibt sich bei dieser Verschlüsselungsform ein 256-Bit (32 Byte) langer FVEK. Dem Programm Dislocker war es mit dem entsprechenden entsiegelten VMK und weiteren Meta-Informationen (Nonce, Input Buffer, MAC) aus der Datensicherung immer möglich den FVEK zu extrahieren. Mit diesem konnte die BitLocker-Partition entschlüsselt werden.

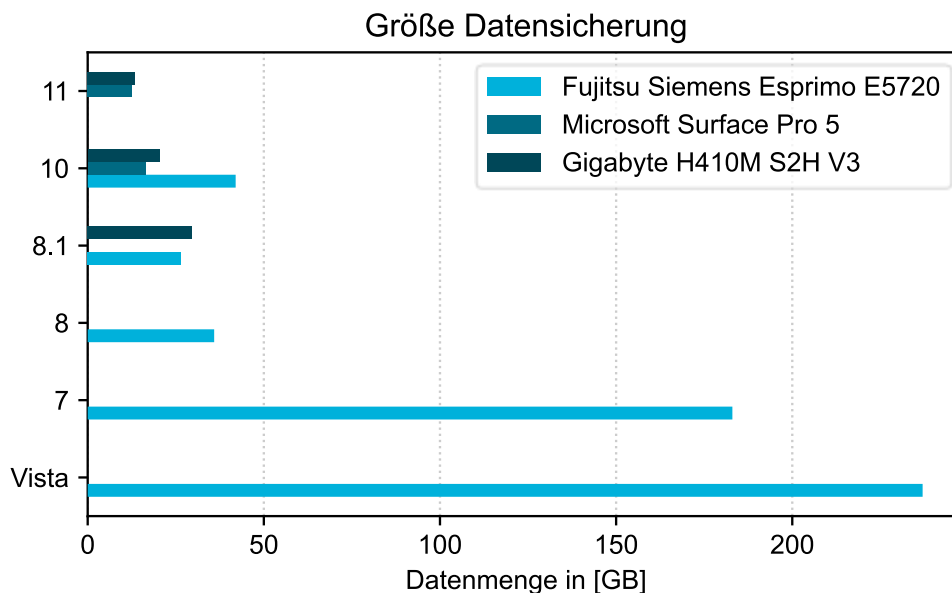
Ein weiterer Unterschied, der bei den Messungen festgestellt wurde, ist die Nutzung der PCRs aus dem TPM zur Validierung des Systems. Die Information zu den genutzten PCRs befinden sich in der Ausgabe des `manage-bde.exe -protectors` Befehls. Hier wurde festgestellt, dass auch bei gleichem Betriebssystem unterschiedliche PCRs genutzt werden. Die Ergebnisse zu den genutzten PCRs und den Zielsystemen sind in **Tabelle 21** dargestellt. Für die Betriebssysteme Vista, 7 und 8 wurden vom BitLocker-Konfigurations-Tool keine Angaben zu den PCRs gemacht. Betriebssysteme, die auf den Zielgeräten nicht installiert werden konnten, sind grau dargestellt.

Betriebssystem	Fujitsu Siemens Esprimo E5720	Microsoft Surface Pro 5	Gigabyte H410M S2H V3
Vista	k.A.		
Win7	k.A.		
Win8	k.A.		
Win8.1	0,2,4,8,9,10,11		0,2,4,11
Win10	0,2,4,8,9,10,11	7,11	0,2,4,11
Win11		7,11	0,2,4,11

**Tabelle 21:** Einfluss Firmware auf PCR Validierungstyp

Die Unterschiede sind durch die Firmware des Zielsystems zu erklären. Bei einer BIOS-Firmware verwendet Windows die PCRs mit den Einträgen 0, 2, 4, 8, 9, 10 und 11. Eine UEFI basierte Firmware verwendet die Einträge 0, 2, 4 und 11. Wie in der **Tabelle 1** im Abschnitt 2.1.1 gezeigt, beinhalten bei einer BIOS-Firmware die PCRs andere Attribute als bei einer UEFI-Firmware. Während bei dem Fujitsu Siemens Esprimo E5720 Zielsystem die PCRs für das CRTM, dem Option-ROM-Code, dem MBR-Code und verschiedene Betriebssystem-Einstellungen abgefragt werden müssen, genügt BitLocker bei einer UEFI-basierten Firmware die Abfrage der PCRs zum nicht-veränderlichen und veränderlichen UEFI-Code, dem Boot Manager und der BitLocker Zugriffssteuerung. Eine weitere Veränderung wurde bei der Verwendung einer UEFI-Firmware in Verbindung mit Secure Boot festgestellt. Hier reduzieren sich die zu überprüfenden PCRs auf die Einträge 7 und 11. Das Register 7 enthält Informationen zur Integrität der Secure Boot Einstellung selbst. Dies kann damit erklärt werden, dass die Secure Boot Einstellung die Integritätsprüfung der PCRs zum nicht-veränderlichen und veränderlichen UEFI-Code und dem Boot Manager bereits beinhaltet. Veränderungen an diesen führen zwangsläufig auch zu Veränderungen an dem Secure Boot Register im TPM. Eine erneute Prüfung dieser Register ist somit nicht notwendig.

Ein weiterer Randeffekt, der untersucht wurde, sind die anfallenden Datenmengen bei diesem Versuch. Daten fallen zum einem mit der Datensicherung und zum anderen durch den Logikanalysator und dem CSV-Datei Export an. Diese beiden Prozesse werden nachfolgend untersucht. Die Datenmengen für die jeweiligen Datensicherung sind in **Abbildung 60** dargestellt.

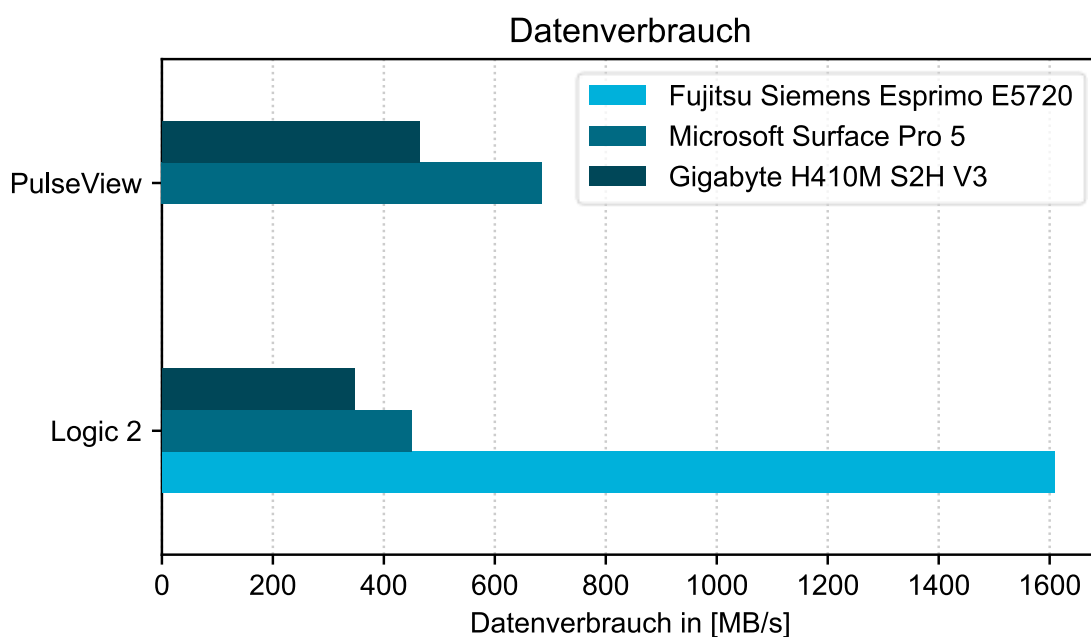


**Abbildung 60:** Vergleich Datensicherungen

Dabei fällt auf, dass die Datensicherungen bei Windows Vista und 7 um ein Vielfaches größer ausfallen als bei neueren Betriebssystemen. Ursache dafür ist, dass erst mit der Einführung der BitLocker Version 2.0 die Option „nur verwendeten Speicherplatz verschlüsseln“ eingeführt worden ist. Diese wird seitdem auch als Standard vorgegeben. Bei älteren Betriebssystemen wurde immer der gesamte Speicherbereich verschlüsselt. Verschlüsselte Daten weisen eine hohe Entropie auf, die kaum komprimiert werden können. Dies führt dazu, dass die Datensicherung im EWF-Format eines vollverschlüsselten Datenträgers in etwa auch der Gesamtkapazität des Datenträgers entspricht. Bei neueren Systemen verfügt nur der verwendete Speicherplatz über eine hohe Entropie der Daten, die kaum komprimiert werden können. Die nicht verwendeten Speicherbereiche, welche mit dem `sdelete64` Befehl durch Nullen

überschrieben wurden, können dagegen sehr gut komprimiert werden und führen zu kleineren Datenmengen. Weiterhin hat sich die Größe der Datensicherungen auch mit den neueren Betriebssystemen reduziert. Ursache hierfür könnte sein, dass bei den älteren Betriebssystemen wesentlich mehr Updates eingespielt werden mussten. Umso länger ein Betriebssystem verfügbar ist, desto mehr sicherheitsrelevante Updates wurden vom Hersteller nachgeliefert. Das Einspielen der Updates nimmt einen nicht unwesentlichen Speicherbereich auf dem Datenträger ein.

Um die anfallenden Datenmengen des Logikanalysators und dem CSV-Datei Export zu bestimmen, wurden die erzeugten CSV-Dateien untersucht. Die Aufnahmedauer hat einen wesentlichen Einfluss auf die Größe der CSV-Datei. Da die Zielsysteme allerdings unterschiedlich lange Startphasen besitzen, wurde für die Auswertung das Verhältnis aus Größe der CSV-Datei und Aufnahmedauer gebildet. Aus den Ergebnissen wurden für jede Logikanalysator Software der Mittelwert gebildet. In der **Abbildung 61** ist dargestellt, wie groß die Datenmenge pro Sekunde Aufnahmedauer ausgefallen ist. Die Kombination aus PulseView und dem Fujitsu Siemens Esprimo E5720 wurde nicht ausgewertet, da aufgrund der zu geringen Abtasttiefe ein Dekodieren der CSV-Datei nicht möglich war.



**Abbildung 61:** Abhängigkeit Datenmenge Logikanalysator

Bei dem Vergleich der beiden Bus-Protokolle SPI und LPC fällt auf, dass die exportierte Datenmenge beim LPC-Bus größer ist. Dies kann mit der Anzahl an aufgenommenen Signalleitungen erklärt werden. Beim SPI-Bus müssen lediglich vier Signalleitungen untersucht werden, während es beim LPC-Bus sieben sind. Obwohl das Microsoft Surface Pro 5 mit dem gleichen Bus-Protokoll (LPC) wie der Fujitsu Siemens Esprimo E5720 kommuniziert, ist die Datenmenge mit 450 MB/s geringer als beim Esprimo (1608 MB/s). Betrachtet man den Bus-Takt beider Geräte so fällt auf, dass dieser beim Fujitsu Siemens Esprimo E5720 durchgehend ist. Beim Microsoft Surface Pro 5 dagegen setzt der Takt aus und wird nur dann aktiviert, wenn eine Kommunikation stattfindet. Das senkt den Energieverbrauch und führt bei der Messung zu weniger anfallenden Daten. Vergleicht man weiterhin die beiden Logikanalysator Programme hinsichtlich ihrer erzeugten Datenmenge so fällt auf, dass bei allen Messungen das Programm Saleae Logic 2 weniger Daten erzeugt hat. Theoretisch sollte die Datenmenge größer sein, weil das Programm Saleae Logic 2 mit einer höheren Abtastrate (125 MS/s) abtastet als PulseView (50 MS/s). Der Grund dafür ist, dass das Programm Saleae Logic 2 nur Datenänderungen auf den Signalleitungen in der Exportdatei erfasst. Aufeinanderfolgende gleiche Daten werden ignoriert. [63] Das führt dazu, dass die Anzahl der Einträge in der CSV-Datei nicht mehr durch die Abtastrate bestimmt wird. Das Resultat sind kleinere Ausgabedateien. PulseView hingegen speichert alle erfassten Daten in eine Datei. Die Größe der CSV-Datei wird hier durch die Abtastrate bestimmt.

## 4 Zusammenfassung

Mit dem in dieser Arbeit vorgestellten Versuch konnte gezeigt werden, dass es mit Hilfe des TPM-Bus-Sniffing Angriffes möglich ist, eine bestehende BitLocker Verschlüsselung anzugreifen. Voraussetzungen hierfür sind, dass der versiegelte VMK mit einem TPM geschützt wird und dieses als eigenes Bauteil auf dem Mainboard realisiert ist. Es wurden die Auswirkungen verschiedener TPMs, Ziel- und Betriebssysteme untersucht, sowie die notwendige Software für den TPM-Bus-Sniffing Angriff programmiert.

Bei Einhaltung bestimmter Randbedingungen ist der Angriff erfolgreich. Als limitierende Randbedingungen dieses Angriffes hat sich die Software zum Dekodieren der Bus-Kommunikation, die Abtastrate und die erzeugte Datenmenge gezeigt. Für den Angriff ist es notwendig die aufgenommene Bus-Kommunikation dekodieren zu können. Als Teil dieser Arbeit konnte erfolgreich ein Programm entwickelt werden, das in der Lage ist, verschiedene Bus-Protokolle zu dekodieren und den entsiegelten VMK zu extrahieren. Dem erfolgreichen Dekodieren setzt allerdings die Wahl der richtigen Abtastfrequenz mit dem Logikanalysator voraus. Wird das Abtasttheorem nicht eingehalten, kann die aufgezeichnete Kommunikation nicht ausgewertet werden. Hinsichtlich der erzeugten Daten, welche für den Angriff verarbeitet werden müssen, betragen diese mehrere Gigabyte und sind abhängig von der Abtastrate und der Aufnahmedauer des Angriffes. Die geringsten Datenmengen wurde mit Logikanalysator Software Saleae Logic 2 erzeugt. Es muss sichergestellt werden, dass ausreichend Speicherplatz auf dem angreifenden System vorhanden ist.

Das genutzte Windows Betriebssystem und der daraus resultierende Standard-Verschlüsselungsalgorithmus haben keinen Einfluss auf den Erfolg des Angriffes, da sich die Schlüsselhierarchie und die Struktur des entsiegelten VMK nicht verändert haben. Die System-Firmware und das Verwenden von Secure Boot als erweitertes Sicherheitskonzept haben Auswirkungen auf die genutzten PCRs von BitLocker. Da bei diesem Angriff allerdings keine für das System erkennbare Manipulation an der Hard- und Software stattfindet, erfolgt keine Veränderung an den PCRs im TPM. Der Angriff bleibt dem System verborgen und ist so wiederholt am System durchführbar.

Die Ursache für den Erfolg des Angriffes beruht auf der fehlenden verschlüsselten Kommunikation zwischen TPM und dem System. Hier bleibt die Frage offen, warum BitLocker keine verschlüsselte Kommunikation mit dem TPM anfordert. Die technischen Voraussetzungen hierfür sind seit der TPM-Spezifikation 2.0 mit der Parameter Encryption gegeben. Microsoft ist sich dieser Schwachstelle bewusst und bietet als Gegenmaßnahme die Kombination aus TPM + PIN und TPM + PIN + USB an. Wird BitLocker aber im Standard belassen, ermöglicht dies einen erfolgreichen TPM-Bus-Sniffing Angriff. Um den Schutz der BitLocker Gegenmaßnahmen zu untersuchen, könnte in einer nachfolgenden Arbeit der TPM-Bus-Sniffing Angriff auf diese Schutzvarianten wiederholt werden. Ein anderer Punkt, der weitergehend untersucht werden kann, ist die Realisierung des TPM als fTPM. In dieser Arbeit wurden die Unterschiede zwischen fTPM und dTPM angedeutet. Diese können weiter untersucht werden, um zu prüfen, ob der TPM-Bus-Sniffing Angriff auch auf ein fTPM übertragen werden kann.

Zusammenfassend lässt sich festhalten, dass der vorgestellte Versuchsaufbau erfolgreich war. Er kann zusammen mit der in dieser Arbeit entwickelten Software für weitere Untersuchungen an BitLocker und anderen TPM basierten Festplattenverschlüsselungen verwendet werden.

## Literaturverzeichnis

- [1] N. Pohlmann, Cyber-Sicherheit - Das Lehrbuch für Konzepte, Prinzipien, Mechanismen, Architekturen und Eigenschaften von CyberSicherheitssystemen in der Digitalisierung, Gelsenkirchen: Springer Vieweg, 2019.
- [2] S. Spitz, M. Pramateftakis und J. Swoboda, Kryptographie und IT-Sicherheit - Grundlagen und Anwendungen, Bde. %1 von %22., überarbeitete Auflage, Wiesbaden: Vieweg+Teubner Verlag, 2011.
- [3] Trusted Computing Group (TCG), „Member Companies | Trusted Computing Group,“ 18. März 2022. [Online]. Available: <https://trustedcomputinggroup.org/membership/member-companies/>. [Zugriff am 13. Mai 2022].
- [4] Trusted Computing Group (TCG), „About TCG | Trusted Computing Group,“ 15 Juli 2021. [Online]. Available: <https://trustedcomputinggroup.org/about>. [Zugriff am 13. Mai 2022].
- [5] Trusted Computing Group, „TCG PC Client Platform TPM Profile Specification for TPM 2.0 (Version 1.05),“ 4 September 2020. [Online]. Available: <https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-tpm-specification/>. [Zugriff am 4. April 2022].
- [6] A. Lunkeit und W. Zimmer, Security by Design - Security Engineering informationstechnischer Systeme, Schildow: Springer Vieweg, 2021.
- [7] Trusted Computing Group (TCG), „Trusted Platform Module Library - Part 1: Architecture,“ 8. November 2019. [Online]. Available: <https://trustedcomputinggroup.org/wp->



- content/uploads/TCG\_TPM2\_r1p59\_Part1\_Architecture\_pub.pdf. [Zugriff am 20. Mai 2022].
- [8] W. Arthur, D. Challenger und K. Goldman, *A Practical Guide to TPM 2.0 - Using the Trusted Plattform Module in the New Age of Security*, Berkeley: Apress, 2015.
- [9] K. D. J. Winter, „A Hijacker’s Guide to the LPC Bus,“ in *Public Key Infrastructures, Services and Applications*, Berlin, Springer-Verlag Berlin Heidelberg, 2012, S. 209.
- [10] Trusted Computing Group (TCG), „TPM Main - Part 1 Design Principles,“ 1. März 2011. [Online]. Available: [https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles\\_v1.2\\_rev116\\_01032011.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf). [Zugriff am 19. Mai 2022].
- [11] Trusted Computing Group, „TCG PC Client Platform - Reset Attack Mitigation Specification,“ 21. Januar 2019. [Online]. Available: [https://trustedcomputinggroup.org/wp-content/uploads/TCG\\_PlatformResetAttackMitigationSpecification\\_1.10\\_published.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TCG_PlatformResetAttackMitigationSpecification_1.10_published.pdf). [Zugriff am 17. Juli 2022].
- [12] Trusted Computing Group (TCG), „Trusted Platform Module Library - Part 3: Command,“ Trusted Computing Group (TCG), 29. September 2016. [Online]. Available: <https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-3-Commands-01.38.pdf>. [Zugriff am 2. September 2022].
- [13] Trusted Computing Group (TCG), „TPM Main - Part 3 Commands,“ Trusted Computing Group (TCG), 1. März 2011. [Online]. Available: [https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-3-Commands\\_v1.2\\_rev116\\_01032011.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-3-Commands_v1.2_rev116_01032011.pdf). [Zugriff am 1. September 2022].

- [14] Microsoft, „Grundlagen zu Trusted Platform Module,“ Microsoft, 13. Juli 2022. [Online]. Available: <https://docs.microsoft.com/de-de/windows/security/information-protection/tpm/tpm-fundamentals#anti-hammering>. [Zugriff am 1. September 2022].
- [15] H. Raj, S. Saroiu, A. Wolman, R. Aigner, J. Cox, P. England, C. Fenner, K. Kinshumann, J. Loeser, D. Mattoon, M. Nystrom, D. Robinson, R. Spiger, S. Thom und D. Wooten, „fTPM: A Software-Only Implementation of a TPM Chip,“ 10. August 2016. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/raj>. [Zugriff am 21. Mai 2022].
- [16] Bundesamt für Sicherheit in der Informationstechnik (BSI), „Kryptographische Verfahren: Empfehlungen und Schlüssellängen,“ 28. Januar 2022. [Online]. Available: [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?\\_\\_blob=publicationFile](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile). [Zugriff am 9. Juni 2022].
- [17] E. Agostini und M. Bernaschi, „BitCracker: BitLocker meets GPUs,“ Springer-Verlag GmbH, Int. J. Inf. Secur., 2022.
- [18] Microsoft, „BitLocker,“ 4. Juni 2022. [Online]. Available: <https://docs.microsoft.com/de-de/windows/security/information-protection/bitlocker/bitlocker-overview>. [Zugriff am 6. Juni 2022].
- [19] Microsoft, „Geräteverschlüsselung in Windows,“ 8. März 2022. [Online]. Available: <https://support.microsoft.com/de-de/windows/ger%C3%A4teverschl%C3%BCsslung-in-windows-ad5dcf4b-dbe0-2331-228f-7925c2a3012d>. [Zugriff am 6. Juni 2022].
- [20] Apple, „Das Startvolume Ihres Mac mit FileVault verschlüsseln,“ [Online]. Available: <https://support.apple.com/de-de/HT204837>. [Zugriff am 3. Juni 2022].

- [21] O. Choudary, F. Grobert und J. Metz, „Infiltrate the Vault: Security Analysis and Decryption of Lion Full Disk Encryption,“ 2012. [Online]. Available: <https://eprint.iacr.org/2012/374.pdf>. [Zugriff am 6. Juni 2022].
- [22] Apple, „Informationen zum verschlüsselten Speicher auf dem neuen Mac,“ [Online]. Available: <https://support.apple.com/de-de/HT208344>. [Zugriff am 3. Juni 2022].
- [23] C. Fruhwirth und M. Schuster, „Festplattenverschlüsselung mit DM-Crypt und Cryptsetup-LUKS: Technik und Anwendung,“ August 2005. [Online]. Available: <https://www.linux-magazin.de/ausgaben/2005/08/geheimeniederschrift/>. [Zugriff am 3. Juni 2022].
- [24] openSUSE, „SDB:LUKS2, TPM2 and FIDO2 - openSUSE Wiki,“ 24. Mai 2022. [Online]. Available: [https://en.opensuse.org/SDB:LUKS2,\\_TPM2\\_and\\_FIDO2](https://en.opensuse.org/SDB:LUKS2,_TPM2_and_FIDO2). [Zugriff am 3. Juni 2022].
- [25] Bundesamt für Sicherheit in der Informationstechnik (BSI), „Security Evaluation of VeraCrypt,“ 30. November 2020. [Online]. Available: [https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Veracrypt/Veracrypt.pdf;jsessionid=01587FCCA1E851C525B756B3876E7E97.internet461?\\_\\_blob=publicationFile&v=1](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Veracrypt/Veracrypt.pdf;jsessionid=01587FCCA1E851C525B756B3876E7E97.internet461?__blob=publicationFile&v=1). [Zugriff am 6. Juni 2022].
- [26] National Institute of Standards and Technology (NIST), „Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices,“ Januar 2010. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38e.pdf>. [Zugriff am 9. Juni 2022].

- [27] P. Shabana Subair, C. Balan, S. Dija und K. Thomas, „Forensic Decryption of FAT BitLocker Volumes,“ Centre for Development of Advanced Computing, Kerala, India, 2014.
- [28] National Institute of Standards and Technology (NIST), „BitLocker™ Drive Encryption Security Policy - For FIPS 140-2 Validation,“ 31. August 2011. [Online]. Available: <https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp1053.pdf>. [Zugriff am 24. Juni 2022].
- [29] J. B. Metz, „libbde/BitLocker Drive Encryption (BDE) format.asciidoc at main · libyal/libbde · GitHub,“ Februar 2022. [Online]. Available: [https://github.com/libyal/libbde/blob/main/documentation/BitLocker%20Drive%20Encryption%20\(BDE\)%20format.asciidoc](https://github.com/libyal/libbde/blob/main/documentation/BitLocker%20Drive%20Encryption%20(BDE)%20format.asciidoc). [Zugriff am 24. Juni 2022].
- [30] Microsoft, „BitLocker-Gegenmaßnahmen,“ 13. Juni 2022. [Online]. Available: <https://docs.microsoft.com/de-de/windows/security/information-protection/bitlocker/bitlocker-countermeasures>. [Zugriff am 24. Juni 2022].
- [31] D. Labudde und M. Spranger, Forensik in der digitalen Welt - Moderne Methoden der forensischen Fallarbeit in der digitalen und digitalisierten realen Welt, Mittweida: Springer Spektrum, 2017.
- [32] J. Yao, V. J. Zimmer und S. Zeng, „A Tour Beyond BIOS: Using IOMMU for DMA Protection in UEFI Firmware,“ 17. Oktober 2017. [Online]. Available: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKewjg9Y3SqID3AhVORfEDHWovCD8QFnoECAUQAQ&url=https%3A%2F%2Fwww.intel.com%2Fcontent%2Fdam%2Fdevelop%2Fexternal%2Fus%2Fen%2Fdocuments%2Fintel-whitepaper-using-iommu-for-dma>. [Zugriff am 6. April 2022].

- [33] U. Frisk, „Direct Memory Attack the KERNEL,“ 16. Mai 2020. [Online]. Available:  
<https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Ulf-Frisk-Direct-Memory-Attack-the-Kernel.pdf>. [Zugriff am 6. April 2022].
- [34] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum und E. W. Felten, „Lest We Remember: Cold Boot Attacks on Encryption Keys,“ USENIX Association, 17th USENIX Security Symposium, 2008.
- [35] J. Bauer, M. Gruhn und F. C. Freiling, „Lest we forget: Cold-boot attacks on scrambled DDR3,“ DFRWS, DFRWS EU 2016, 2016.
- [36] D. Andzakovic, „Extracting BitLocker keys from a TPM,“ Pulse Security, 13. März 2019. [Online]. Available:  
<https://pulsesecurity.co.nz/articles/TPM-sniffing>. [Zugriff am 1. Juli 2022].
- [37] Intel, „Intel® Chipsets Low Pin Count Interface Specification - low-pin-count-interface-specification.pdf,“ 1 August 2002. [Online]. Available:  
<https://www.intel.de/content/dam/www/program/design/us/en/documents/low-pin-count-interface-specification.pdf>. [Zugriff am 24 März 2022].
- [38] S. C. Hill, J. Jelemensky und M. R. Heene, „Queued serial peripheral interface for use in a data processing system,“ 24. Juli 1987. [Online]. Available: <https://patents.google.com/patent/US4816996A/en>. [Zugriff am 14. April 2022].
- [39] Intel, „Enhanced Serial Peripheral Interface (eSPI),“ 1. Januar 2016. [Online]. Available:  
[https://www.intel.com/content/dam/support/us/en/documents/software/chipset-software/327432-004\\_espi\\_base\\_specification\\_rev1.0\\_cb.pdf](https://www.intel.com/content/dam/support/us/en/documents/software/chipset-software/327432-004_espi_base_specification_rev1.0_cb.pdf). [Zugriff am 14. April 2022].

- [40] H. Bernstein, Mikrocontroller, Grundlagen der Hard- und Software der Mikrocontroller ATiny2313, ATiny26 und ATmega32, 2., aktualisierte und erweiterte Auflage Hrsg., München: Springer Vieweg, 2020.
- [41] A. Meroth und P. Sora, Sensornetzwerke in Theorie und Praxis, 2. Auflage Hrsg., Wiesbaden: Springer Vieweg, 2021.
- [42] P. Dhaker, „Introduction to SPI Interface,“ 9. September 2018. [Online]. Available: <https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html>. [Zugriff am 15. April 2022].
- [43] Microsoft, „BitLocker recovery password details,“ 10. August 2006. [Online]. Available: [https://docs.microsoft.com/de-de/archive/blogs/si\\_team/bitlocker-recovery-password-details](https://docs.microsoft.com/de-de/archive/blogs/si_team/bitlocker-recovery-password-details). [Zugriff am 1. Juli 2022].
- [44] S. Boblest, T. Müller und G. Wunner, Spezielle und allgemeine Relativitätstheorie - Grundlagen, Anwendungen in Astrophysik und Kosmologie sowie relativistische Visualisierung, 2. Auflage Hrsg., Stuttgart: Springer Spektrum, 2021.
- [45] Intel, „2nd Generation Intel® Core™ Processor Family Desktop, Intel® Pentium® Processor Family Desktop, and Intel® Celeron® Processor Family Desktop,“ Juni 2013. [Online]. Available: <https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/2nd-gen-core-desktop-vol-1-datasheet.pdf>. [Zugriff am 6. Juli 2022].
- [46] AMD, „AMD Memory Guard,“ 2020. [Online]. Available: <https://www.amd.com/system/files/documents/amd-memory-guard-white-paper.pdf>.
- [47] J.-C. Delaunay, „IOMMU and DMA attacks,“ 20. November 2019. [Online]. Available:

- [https://www.synacktiv.com/ressources/IOMMU\\_and\\_DMA\\_attacks\\_presentation\\_16\\_9.pdf](https://www.synacktiv.com/ressources/IOMMU_and_DMA_attacks_presentation_16_9.pdf). [Zugriff am 6. April 2022].
- [48] Fujitsu Siemens Computers, „ESPRIMO E Green PC Small Formfaktor PC - Datenblatt,“ Fujitsu Siemens Computers, 2008.
- [49] Microsoft, „Surface Pro 2-Features,“ 2. Dezember 2019. [Online]. Available: <https://support.microsoft.com/de-de/surface/features-von-surface-pro-5-gen-42d321e4-52d6-dcb1-e014-9ffc76fbca14>. [Zugriff am 18. Juli 2022].
- [50] R. Wilkins und B. Richardson, „UEFI Secure Boot In Modern,“ September 2013. [Online]. Available: [https://uefi.org/sites/default/files/resources/UEFI\\_Secure\\_Boot\\_in\\_Modern\\_Computer\\_Security\\_Solutions\\_2013.pdf](https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf). [Zugriff am 17. September 2022].
- [51] Microsoft, „Windows Vista - Microsoft Lifecycle,“ Microsoft, 1. September 2020. [Online]. Available: <https://docs.microsoft.com/de-de/lifecycle/products/windows-vista>. [Zugriff am 22. Juli 2022].
- [52] Microsoft, „Windows 7 - Microsoft Lifecycle,“ Microsoft, 19. Oktober 2021. [Online]. Available: <https://docs.microsoft.com/de-de/lifecycle/products/windows-7>. [Zugriff am 22. Juli 2022].
- [53] Microsoft, „Windows 8 - Microsoft Lifecycle,“ Microsoft, 9. Oktober 2020. [Online]. Available: <https://docs.microsoft.com/de-de/lifecycle/products/windows-8>. [Zugriff am 22. Juli 2022].
- [54] Microsoft, „Windows 8.1 - Microsoft Lifecycle,“ Microsoft, 9. Oktober 2020. [Online]. Available: <https://docs.microsoft.com/de-de/lifecycle/products/windows-81>. [Zugriff am 22 Juli 2022].
- [55] Microsoft, „Windows 10 Home und Pro - Microsoft Lifecycle,“ Microsoft, 15. November 2021. [Online]. Available: <https://docs.microsoft.com/de-de>

- de/lifecycle/products/windows-10-home-and-pro. [Zugriff am 22. Juli 2022].
- [56] Microsoft, „Windows 11 Home and Pro (Version 21H2) - Microsoft Lifecycle,“ Microsoft, 13. Oktober 2022. [Online]. Available: <https://docs.microsoft.com/de-de/lifecycle/products/windows-11-home-and-pro-version-21h2?branch=live>. [Zugriff am 22. Juli 2022].
- [57] Microsoft, „Surface-unterstützte Betriebssysteme,“ Microsoft, 15. Januar 2021. [Online]. Available: [https://support.microsoft.com/de-de/surface/von-surface-unterst%C3%BCtzte-betriebssysteme-9559cc3c-7a38-31b6-d9fb-571435e84cd1#bkmk\\_surfacepro](https://support.microsoft.com/de-de/surface/von-surface-unterst%C3%BCtzte-betriebssysteme-9559cc3c-7a38-31b6-d9fb-571435e84cd1#bkmk_surfacepro). [Zugriff am 2. August 2022].
- [58] Gigabyte, „H410M S2H V3 (rev. 1.0) Support | Motherboard - GIGABYTE Global,“ Gigabyte, [Online]. Available: <https://www.gigabyte.com/Motherboard/H410M-S2H-V3-rev-10/support#support-dl>. [Zugriff am 5. August 2022].
- [59] Microsoft, „Windows 11-Spezifikationen, -Funktionen und -Computeranforderungen,“ Microsoft, [Online]. Available: <https://www.microsoft.com/de-de/windows/windows-11-specifications>. [Zugriff am 31. August 2022].
- [60] M. Meyer, Signalverarbeitung, 8 Hrsg., Windisch: Springer Vieweg Wiesbaden, 2017, S. 324.
- [61] Microsoft, „SDelete v2.04,“ Microsoft, 7. Juli 2022. [Online]. Available: <https://docs.microsoft.com/en-us/sysinternals/downloads/sdelete>. [Zugriff am 15. September 2022].
- [62] GigaDevice, „GD25B127D - Datasheet,“ [Online]. Available: [www.gigadevice.com/datasheet/gd25b127d/](http://www.gigadevice.com/datasheet/gd25b127d/). [Zugriff am 25. September 2022].



- [63] Saleae, „Exporting Data - Saleae Support,“ Saleae, [Online]. Available: <https://support.saleae.com/user-guide/using-logic/exporting-data>. [Zugriff am 14. September 2022].
- [64] Bundesamt für Sicherheit in der Informationstechnik (BSI), „Sichere Nutzung von Geräten unter Microsoft Windows 10,“ 18. Juli 2017. [Online]. Available: [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/digitaler\\_Verbraucherschutz/Publikationen/BSI\\_CS\\_019.pdf?\\_\\_blob=publicationFile&v=1](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/digitaler_Verbraucherschutz/Publikationen/BSI_CS_019.pdf?__blob=publicationFile&v=1). [Zugriff am 31. März 2022].
- [65] Bundesamt für Sicherheit in der Informationstechnik (BSI), „Sichere Passwörter,“ [Online]. Available: [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Checklisten/sichere\\_passwoerter\\_faktenblatt.pdf?\\_\\_blob=publicationFile&v=1](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Checklisten/sichere_passwoerter_faktenblatt.pdf?__blob=publicationFile&v=1). [Zugriff am 31. März 2022].
- [66] Bundesamt für Sicherheit in der Informationstechnik (BSI), „Konfigurationsempfehlung zur Härtung von Windows 10 mit Boardmitteln,“ [Online]. Available: [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/SiSyPHus/Konfigurationsempfehlungen\\_zur\\_Haertung\\_von\\_Windows\\_10.pdf?\\_\\_blob=publicationFile&v=3](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/SiSyPHus/Konfigurationsempfehlungen_zur_Haertung_von_Windows_10.pdf?__blob=publicationFile&v=3). [Zugriff am 31. März 2022].
- [67] Microsoft, „wscript,“ 3. März 2021. [Online]. Available: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/wscript>. [Zugriff am 4. April 2022].
- [68] Microsoft, „Powershell 7.2,“ 16. Februar 2022. [Online]. Available: <https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.2>. [Zugriff am 4. April 2022].

- [69] Microsoft, „What’s New in Powershell 7.3,“ 2. April 2022. [Online]. Available: <https://docs.microsoft.com/en-us/powershell/scripting/whats-new/what-s-new-in-powershell-73?view=powershell-7.2>. [Zugriff am 4. April 2022].
- [70] National Security Agency (NSA), „UEFI Secure Boot Customization (1.1),“ 1. September 2020. [Online]. Available: <https://media.defense.gov/2020/Sep/15/2002497594/-1/-1/0/CTR-UEFI-Secure-Boot-Customization-UOO168873-20.PDF>. [Zugriff am 5. April 2022].
- [71] Microsoft, „Kernel-DMA-Schutz (Windows) - Windows security,“ 1. April 2022. [Online]. Available: <https://docs.microsoft.com/de-de/windows/security/information-protection/kernel-dma-protection-for-thunderbolt>. [Zugriff am 6. April 2022].
- [72] G. Piolle, „Trusted Platform Module – Wikipedia,“ 18. September 2008. [Online]. Available: [https://de.wikipedia.org/wiki/Trusted\\_Platform\\_Module#/media/Datei:TPM.svg](https://de.wikipedia.org/wiki/Trusted_Platform_Module#/media/Datei:TPM.svg). [Zugriff am 15. Mai 2022].
- [73] Trusted Computing Group (TCG), „TCG TSS 2.0 Enhanced System API (ESAPI) Specification,“ 1. Oktober 2021. [Online]. Available: [https://trustedcomputinggroup.org/wp-content/uploads/TSS\\_ESAPI\\_v1p0\\_r14\\_pub10012021.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TSS_ESAPI_v1p0_r14_pub10012021.pdf). [Zugriff am 28. Juni 2022].
- [74] „statcounter.com,“ [Online]. Available: <https://gs.statcounter.com/os-market-share/desktop/worldwide>. [Zugriff am 21. November 2021].



## Abbildungsverzeichnis

<b>Abbildung 1:</b> Aufbau eines TPM (angelehnt an [2, S. 236]).....	4
<b>Abbildung 2:</b> Integritätsmessung am Beispiel einer BIOS-Firmware (angelehnt an [6, S. 236]).....	9
<b>Abbildung 3:</b> TPM Schlüssehierarchie.....	10
<b>Abbildung 4:</b> 28-Pin TSSOP .....	16
<b>Abbildung 5:</b> 32-Pin QFN.....	16
<b>Abbildung 6:</b> BitLocker Systemstart.....	22
<b>Abbildung 7:</b> BitLocker Angriffsvektoren.....	28
<b>Abbildung 8:</b> Cold Boot - eingefrorener DDR2 Speicher .....	31
<b>Abbildung 9:</b> LPC-Frame – E/A Lese-Zyklus .....	36
<b>Abbildung 10:</b> LPC-Frame, E/A Schreib-Zyklus.....	36
<b>Abbildung 11:</b> eSPI-Bus mit Standard E/A .....	42
<b>Abbildung 12:</b> SPI-Bit-Protokoll .....	47
<b>Abbildung 13:</b> Fujitsu Siemens Esprimo E5720.....	57
<b>Abbildung 14:</b> Mainboard D2594 .....	58
<b>Abbildung 15:</b> Infineon SLB9635TT12.....	58
<b>Abbildung 16:</b> Microsoft Surface Pro 5 .....	58
<b>Abbildung 17:</b> Mainboard Surface Pro 5.....	59
<b>Abbildung 18:</b> Nuvoton NPCT650 .....	59
<b>Abbildung 19:</b> Gigabyte H410M S2H V3 .....	60
<b>Abbildung 20:</b> GC-TPM2.0 SPI 2.0.....	61
<b>Abbildung 21:</b> Zeitsignal und Frequenzspektrum des Bus-Taktes.....	69
<b>Abbildung 22:</b> Einfluss des Betriebssystem auf den Bus-Takt.....	70
<b>Abbildung 23:</b> Programmstart ARNE .....	72
<b>Abbildung 24:</b> Ablaufdiagramm main.py .....	73
<b>Abbildung 25:</b> main.py .....	74
<b>Abbildung 26:</b> main.py (Zeile 22).....	75
<b>Abbildung 27:</b> main.py (Zeile 33).....	75
<b>Abbildung 28:</b> main.py (Zeile 173-179).....	75
<b>Abbildung 29:</b> Zustandsautomat lpc_decoder.py.....	76

<b>Abbildung 30:</b> lpc_decoder.py (Zeile 55-74).....	77
<b>Abbildung 31:</b> lpc_decoder.py (Zeile 78-88).....	78
<b>Abbildung 32:</b> lpc_decoder.py (Zeile 90-103).....	78
<b>Abbildung 33:</b> lpc_decoder.py (Zeile 106-119).....	79
<b>Abbildung 34:</b> lpy_decoder.py (Zeile 121-131).....	79
<b>Abbildung 35:</b> lpc_decoder.py (Zeile 133-144).....	80
<b>Abbildung 36:</b> Zustandsautomat spi_decoder.py.....	80
<b>Abbildung 37:</b> spi_decoder.py (Zeile 64-79).....	81
<b>Abbildung 38:</b> spi_decoder.py (Zeile 81-85).....	81
<b>Abbildung 39:</b> spi_decoder.py (Zeile 87-97).....	82
<b>Abbildung 40:</b> spi_decoder.py (Zeile 99-107).....	82
<b>Abbildung 41:</b> spi_decoder.py (Zeile 109-117).....	83
<b>Abbildung 42:</b> spi_decoder.py (Zeile 119-129).....	84
<b>Abbildung 43:</b> Versuchsaufbau.....	85
<b>Abbildung 44:</b> Windows 8.1 Installation.....	88
<b>Abbildung 45:</b> manage-bde.exe -status (Esprimo Windows 8.1).....	89
<b>Abbildung 46:</b> manage-bde.exe -protecors -get C: (Esprimo Windows 8.1)...	89
<b>Abbildung 47:</b> Ausgabe sdelete64.exe.....	90
<b>Abbildung 48:</b> FTK Imager.....	91
<b>Abbildung 49:</b> TPM Pinout nach TCG-Spezifikation (angelehnt an [5]).....	92
<b>Abbildung 50:</b> Kontaktierung des Microsoft Surface Pro 5.....	93
<b>Abbildung 51:</b> Pinout GigaDevice 25B127D [62].....	94
<b>Abbildung 52:</b> Kontaktierung Gigabyte H410M S2H V3 mit Klemmen am Firmware-Chip.....	94
<b>Abbildung 53:</b> Saleae Logic 2.....	95
<b>Abbildung 54:</b> PulseView.....	96
<b>Abbildung 55:</b> Inhalt Gigabyte Win 11 HLA CSV.....	96
<b>Abbildung 56:</b> ARNE.....	97
<b>Abbildung 57:</b> XMOUNT.....	98
<b>Abbildung 58:</b> losetup.....	98
<b>Abbildung 59:</b> Dislocker.....	99
<b>Abbildung 60:</b> Vergleich Datensicherungen.....	107

**Abbildung 61:** Abhängigkeit Datenmenge Logikanalysator ..... 108

## Tabellenverzeichnis

<b>Tabelle 1:</b> Vergleich PCR Register im BIOS- und UEFI-Firmware [8, S. 152] ...	8
<b>Tabelle 2:</b> Vergleich Verschlüsselungssoftware .....	20
<b>Tabelle 3:</b> Aufbau entsiegelter VMK .....	26
<b>Tabelle 4:</b> Signalleitungen LPC, einfache Ausführung [37, S. 9] .....	35
<b>Tabelle 5:</b> LPC-Zyklus, START-Phase [37, S. 15] .....	37
<b>Tabelle 6:</b> LPC-Zyklus, Cycle Type/Direction-Phase [37, S. 15] .....	38
<b>Tabelle 7:</b> LPC-Zyklus, SYNC [37, S. 17] .....	39
<b>Tabelle 8:</b> eSPI-Signalleitungen .....	43
<b>Tabelle 9:</b> SPI Konfigurationsregister (SPCR) .....	45
<b>Tabelle 10:</b> SPI Modus .....	46
<b>Tabelle 11:</b> John The Ripper – BitLocker Modus Performance .....	51
<b>Tabelle 12:</b> Gegenüberstellung der Angriffsvektoren.....	55
<b>Tabelle 13:</b> TPM Firmware Einstellungen .....	62
<b>Tabelle 14:</b> Gegenüberstellung Zielsysteme.....	64
<b>Tabelle 15:</b> Windows Produkt Lebenszyklus [51] [52] [53] [54] [55] [56].....	65
<b>Tabelle 16:</b> Installierbare Betriebssysteme auf den Zielsystemen .....	65
<b>Tabelle 17:</b> Systemstartzeit .....	71
<b>Tabelle 18:</b> Veränderliche Parameter .....	99
<b>Tabelle 19:</b> Einfluss Verschlüsselungsalgorithmus auf entsiegelten VMK.....	103
<b>Tabelle 20:</b> Einfluss Betriebssystem auf BitLocker .....	104
<b>Tabelle 21:</b> Einfluss Firmware auf PCR Validierungstyp .....	106





## Abkürzungsverzeichnis

AIK	Attestation Key
ARNE	Advanced Re-Engineering of Network-Interception Encryption
BMC	Baseboard Management Controller
CPHA	SPI Clock Phase
CPOL	SPI Idle Polarity
CPU	Central Processing Unit
CRTM	Core Root of Trust Measurement
CS#	SPI Chip Select
CSV	Comma-separated values
DMA	Direct Memory Access
DORD	SPI Data Order
DRAM	Dynamic Random Access Memory
dTPM	diskretes TPM
E/A	Eingabe/Ausgabe
EC	Embedded Controller
ECC	Elliptic Curve Cryptography
EK	Endorsement Key
eSPI	Enhances Serial Peripheral Interface
EWf	Expert Witness Disk Image Format
fTPM	Firmware TPM
FVEK	Full Volume Encryption Key

FWH	Firmware Hub
GND	ground / Masse
I2C	Inter-Integrated-Circuit
IOMMU	I/O memory managment unit
ISA	Industry Standard Architecture
LCLK	LPC Bus-Takt
LPC	Low-Pin-Count
lsb	Least Significant Bit first
MISO	Maset-Input-Slave-Output
MMIO	Memory Mapped I/O
MOR	Memory Overwrite Request
MOSI	Master-Output-Slave-Input
msb	Most Significant Bit first
MSTR	SPI Master/Slave Select
NIST	National Institute of Standards and Technology
NV-RAM	Non-Volatile Random-Access Memory
PBKDF2	Password-Based Key Derivation Function 2
PCH	Plattform Controller Hub
PCI	Peripheral Component Interconnect
PCR	Plattform Configuration Register
PIM	Personal Iterations Multipler
PTT	(Intel) Platform Trust Technology
QFN	Quad Flat No Leads Package
RAM	Random-Access Memory
ROM	Read-Only Memory

RSA	Rivest-Shamir-Adleman Verfahren
SCLK	SPI Bus-Takt
SIO	Super I/O Controller
SPCR	SPI Konfigurationsregister
SPDR	SPI Datenregister
SPI	Serial Peripheral Interface
SPSR	SPI Statusregister
SRK	Storage Root Key
TAR	Turnaround Phase
TCG	Trusted Computing Group
TCPA	Trusted Computing Platform Alliance
TIS	TPM Interface Standard
TNC	Trusted Network Communications
TPM	Trusted Platform Modul
TSSOP	Thin Shrink Small Outline Package
VMK	Volume Master Key



## Digitales Anlagenverzeichnis

Dieser Arbeit ist ein Datenträger beigelegt. Dieses digitale Anlagenverzeichnis führt alle auf dem Datenträger befindlichen Dateien auf.

<b>Ordner</b>	<b>Inhalt</b>
brute-force-messung	Enthält die Ausgaben von John The Ripper um die Geschwindigkeit des Brute-Force Angriffes gegen den Wiederherstellungsschlüssel zu bestimmen
datensicherung	Beinhaltet die Datensicherung des Zielsystems im EFW-Format
bitlocker	Die Wiederherstellungsschlüssel zu den jeweiligen BitLocker-Partitionen sind in diesem Ordner hinterlegt
clk_messung	Die mit einem Oszilloskop aufgenommenen Messdaten sind in Textdateien im Ordner clk_messung abgelegt
boot_messung	Enthält die von der Windows Eventanzeige aufgenommene Messung zur Dauer des Startprozesses

logikanalysator	Beinhaltet die gespeicherten Sitzungen der Logikanalysator Software Logic 2 und PulseView. Zusätzlich sind die Exportierten CSV-Dateien hier abgelegt
ARNE	Hier sind die Ausgaben des Programms ARNE gespeichert, sowie die Python Skripte für das Programm
dislocker	Der Ordner dislocker beinhaltet die Ausgaben von Dislocker, die den FVEK beinhalten
manage-bde	Enthält die Ausgaben des manage-bde Tools mit Informationen über BitLocker

---

Datei	Pfad
MT_SebastianLasogga_352220.pdf	/
ARNE-LINUX.dd	/
jtr-bitlocker-750ti.txt	/brute-force_messung/
jtr-bitlocker-1060.txt	/brute-force_messung/
jtr-bitlocker-1080.txt	/brute-force_messung/
jtr-bitlocker-2080ti.txt	/brute-force_messung/
jtr-bitlocker-3080.txt	/brute-force_messung/
jtr-bitlocker-3080ti.txt	/brute-force_messung/
Esprimo_win-vista.E01	/datensicherung/
Esprimo_win7.E01	/datensicherung/
Esprimo_win8.E01	/datensicherung/
Esprimo_win8-1.E01	/datensicherung/
Esprimo_win10.E01	/datensicherung/
surface-secure-boot_win10.E01	/datensicherung/
surface-secure-boot_win11.E01	/datensicherung/

---

---

Gigabyte_win8-1.E01	/datensicherung/
Gigabyte_win10.E01	/datensicherung/
Gigabyte_win11.E01	/datensicherung/
Esprimo_win-vista_ws.txt	/datensicherung/
Esprimo_win7_ws.txt	/bitlocker/
Esprimo_win8_ws.txt	/bitlocker/
Esprimo_win8-1_ws.txt	/bitlocker/
Esprimo_win10_ws.txt	/bitlocker/
surface-secure-boot_win10_ws.txt	/bitlocker/
surface-secure-boot_win11_ws.txt	/bitlocker/
Gigabyte_win8-1_ws.txt	/bitlocker/
Gigabyte_win10_ws.txt	/bitlocker/
Gigabyte_win11_ws.txt	/bitlocker/
Esprimo_win-vista_clk.txt	/clk_messung/
Esprimo_win7_clk.txt	/clk_messung/
Esprimo_win8_clk.txt	/clk_messung/
Esprimo_win8-1_clk.txt	/clk_messung/
Esprimo_win10_clk.txt	/clk_messung/
Surface_win10_clk.txt	/clk_messung/
Surface_win11_clk.txt	/clk_messung/
Gigabyte_win8-1_clk.txt	/clk_messung/
Gigabyte_win10_clk.txt	/clk_messung/
Gigabyte_win11_clk.txt	/clk_messung/
Esprimo_win-vista_bootzeit.evtx	/boot_messung/
Esprimo_win7_bootzeit.evtx	/boot_messung/
Esprimo_win8_bootzeit.evtx	/boot_messung/
Esprimo_win8-1_bootzeit.evtx	/boot_messung/
Esprimo_win10_bootzeit.evtx	/boot_messung/
Surface_win10_bootzeit.evtx	/boot_messung/
Surface_win11_bootzeit.evtx	/boot_messung/
Gigabyte_win8-1_bootzeit.evtx	/boot_messung/

---

Gigabyte_win10_bootzeit.evtx	/boot_messung/
Gigabyte_win11_bootzeit.evtx	/boot_messung/
Esprimo_win-vista.sr	/logikanalysator/
Esprimo_win7.sr	/logikanalysator/
Esprimo_win8.sr	/logikanalysator/
Esprimo_win8-1.sr	/logikanalysator/
Esprimo_win10.sr	/logikanalysator/
Surface_win10.sr	/logikanalysator/
Surface_win11.sr	/logikanalysator/
Gigabyte_win8-1.sr	/logikanalysator/
Gigabyte_win10.sr	/logikanalysator/
Gigabyte_win11.sr	/logikanalysator/
Esprimo_win-vista.sal	/logikanalysator/
Esprimo_win7.sal	/logikanalysator/
Esprimo_win8.sal	/logikanalysator/
Esprimo_win8-1.sal	/logikanalysator/
Esprimo_win10.sal	/logikanalysator/
Surface_win10.sal	/logikanalysator/
Surface_win11.sal	/logikanalysator/
Gigabyte_win8-1.sal	/logikanalysator/
Gigabyte_win10.sal	/logikanalysator/
Gigabyte_win11.sal	/logikanalysator/
Esprimo_win-vista_saleae.csv	/logikanalysator/
Esprimo_win7_saleae.csv	/logikanalysator/
Esprimo_win8_saleae.csv	/logikanalysator/
Esprimo_win8-1_saleae.csv	/logikanalysator/
Esprimo_win10_saleae.csv	/logikanalysator/
Surface_win10_saleae.csv	/logikanalysator/
Surface_win11_saleae.csv	/logikanalysator/
Gigabyte_win8-1_saleae.csv	/logikanalysator/
Gigabyte_win10_saleae.csv	/logikanalysator/



Gigabyte_win11_saleae.csv	/logikanalysator/
Esprimo_win10_pulseview.csv	/logikanalysator/
Surface_win10_pulseview.csv	/logikanalysator/
Surface_win11_pulseview.csv	/logikanalysator/
Gigabyte_win8-1_pulseview.csv	/logikanalysator/
Gigabyte_win10_pulseview.csv	/logikanalysator/
Gigabyte_win11_pulseview.csv	/logikanalysator/
Esprimo_win-vista_ARNE.txt	/ARNE/
Esprimo_win7_ARNE.txt	/ARNE/
Esprimo_win8_ARNE.txt	/ARNE/
Esprimo_win8-1_ARNE.txt	/ARNE/
Esprimo_win10_ARNE.txt	/ARNE/
Surface_win10_ARNE.txt	/ARNE/
Surface_win11_ARNE.txt	/ARNE/
Gigabyte_win8-1_ARNE.txt	/ARNE/
Gigabyte_win10_ARNE.txt	/ARNE/
Gigabyte_win11_ARNE.txt	/ARNE/
Esprimo_win-vista.vmk	/ARNE/
Esprimo_win7.vmk	/ARNE/
Esprimo_win8.vmk	/ARNE/
Esprimo_win8-1.vmk	/ARNE/
Esprimo_win10.vmk	/ARNE/
Surface_win10.vmk	/ARNE/
Surface_win11.vmk	/ARNE/
Gigabyte_win8-1.vmk	/ARNE/
Gigabyte_win10.vmk	/ARNE/
Gigabyte_win11.vmk	/ARNE/
main.py	/ARNE/Skript/
var.py	/ARNE/Skript/
lpc_decoder.py	/ARNE/Skript/
spi_decoder.py	/ARNE/Skript/

Esprimo_win-vista_dislocker.txt	/dislocker/
Esprimo_win7_dislocker.txt	/dislocker/
Esprimo_win8_dislocker.txt	/dislocker/
Esprimo_win8-1_dislocker.txt	/dislocker/
Esprimo_win10_dislocker.txt	/dislocker/
Surface_win10_dislocker.txt	/dislocker/
Surface_win11_dislocker.txt	/dislocker/
Gigabyte_win8-1_dislocker.txt	/dislocker/
Gigabyte_win10_dislocker.txt	/dislocker/
Gigabyte_win11_dislocker.txt	/dislocker/
Esprimo_win-vista_manage-bde.txt	/manage-bde/
Esprimo_win7_manage-bde.txt	/manage-bde/
Esprimo_win8_manage-bde.txt	/manage-bde/
Esprimo_win8-1_manage-bde.txt	/manage-bde/
Esprimo_win10_manage-bde.txt	/manage-bde/
Surface_win10_manage-bde.txt	/manage-bde/
Surface_win11_manage-bde.txt	/manage-bde/
Gigabyte_win8-1_manage-bde.txt	/manage-bde/
Gigabyte_win10_manage-bde.txt	/manage-bde/
Gigabyte_win11_manage-bde.txt	/manage-bde/
Esprimo_win-vista_manage-bde-protectors.txt	/manage-bde/
Esprimo_win7_manage-bde-protectors.txt	/manage-bde/
Esprimo_win8_manage-bde-protectors.txt	/manage-bde/
Esprimo_win8-1_manage-bde-protectors.txt	/manage-bde/
Esprimo_win10_manage-bde-protectors.txt	/manage-bde/
Surface_win10_manage-bde-protectors.txt	/manage-bde/
Surface_win11_manage-bde-protectors.txt	/manage-bde/
Gigabyte_win8-1_manage-bde-protectors.txt	/manage-bde/
Gigabyte_win10_manage-bde-protectors.txt	/manage-bde/
Gigabyte_win11_manage-bde-protectors.txt	/manage-bde/
Esprimo_win-vista_systeminfo.txt	/systeminfo/

Esprimo_win7_systeminfo.txt	/systeminfo/
Esprimo_win8_systeminfo.txt	/systeminfo/
Esprimo_win8-1_systeminfo.txt	/systeminfo/
Esprimo_win10_systeminfo.txt	/systeminfo/
Surface_win10_systeminfo.txt	/systeminfo/
Surface_win11_systeminfo.txt	/systeminfo/
Gigabyte_win8-1_systeminfo.txt	/systeminfo/
Gigabyte_win10_systeminfo.txt	/systeminfo/
Gigabyte_win11_systeminfo.txt	/systeminfo/



## **Selbständigkeitserklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Arbeit entspricht der elektronischen Fassung. Ich stimme zu, dass eine elektronische Kopie gefertigt und gespeichert werden darf, um eine Überprüfung mittels Anti-Plagiatssoftware zu ermöglichen.

Berlin, 07.09.2022

Ort, Datum

\_\_\_\_\_  
(Unterschrift)