der Bundeswehr
Universität München
FAKULTÄT FÜR INFORMATIK

Sichere und integre IT-Systeme

Dr. Udo Helmbrecht
Prof. Dr. Gabi Dreo Rodosek
Björn Stelte
Robert Koch
(Hrsg.)

Institut für Technische Informatik

Bericht 2009-02
März 2009

Werner-Heisenberg-Weg 39 • 85577 Neubiberg • Germany

Vorwort

Wir leben heute in einer globalen Informationsgesellschaft. Die Informationstechnik hat inzwischen alle Bereiche des gesellschaftlichen Lebens durchdrungen und ist ein selbstverständlicher Bestandteil des Alltags geworden. IT-Systeme steuern nicht nur Produktionsprozesse und den elektronischen Geschäftsverkehr - Miniaturisierung, Vernetzung und Einbettung von IT sind die Kennzeichnen des Pervasive Computing. Damit ist Informationstechnik heute gleichermaßen allgegenwärtig sowie unsichtbar geworden: Mikrocomputer sind in Alltagsgegenstände integriert und machen diese zu intelligenten Gegenständen, die für ihre Benutzer zahlreiche Aufgaben übernehmen. So können z.B. Fahrassistenzsysteme heute in Extremsituationen sogar menschliche Schwächen ausgleichen. Den zahlreichen Vorteilen des stark verbreiteten Einsatzes von Informationstechnik stehen auch einige Nachteile in Form von Sicherheitsrisiken gegenüber. Über 40 Millionen Menschen sind allein in Deutschland online. Die weltweite Verfügbarkeit des Internets stellt zunehmend ein lohnenswertes Ziel für Angriffe auf die Vertraulichkeit, Verfügbarkeit und Integrität von Daten dar. Die Online-Kriminalität agiert bereits heute in Form einer florierenden Untergrundwirtschaft. Die Angreifer arbeiten international, arbeitsteilig und hoch professionell. Um sich vor der steigenden Anzahl von Bedrohungen durch Schadprogramme zu schützen, benötigen wir sichere und integre IT-Systeme. Trustet Computing und sichere Plattformen können einen wichtigen Beitrag zur Absicherung der Systeme leisten. Doch nur wenn Sicherheit, Funktionalität und Leistungsfähigkeit als gleichrangige Ziele bei der Entwicklung und beim Erwerb von Rechnersystemen anerkannt werden, kann es eine sichere und vertrauenswürdige Nutzung von Informationstechnik geben.
Die Beiträge dieses Seminarbandes geben einen Überblick über die Bedrohungsszenarien und möglichen Lösungsansätze aus den Bereichen Kryptographie, Trusted Computing und des Autonomic Computing.

Udo Helmbrecht
Februar 2009

Inhaltsverzeichnis

Kapitel 1

Diffie - Hellman - Kryptographie mit Hilfe elliptischer Kurven

Keno Dinkela

Sichere und vertrauenswürdige Kommunikation ist im Internet nur durch den Einsatz kryptographischer Verfahren möglich. Beispiel sind die SSL-Verschlüsselung, digitale Signaturen, Sprach- und Datenverschlüsselung. Hinreichende Sicherheit (Integrität und Vertraulichkeit der Daten) und eine aussagekräftige Authentizität (Identifikation und Unabstreitbarkeit) können auf der Grundlage einer Public Key Infrastruktur (PKI) erreicht werden. Mit Hilfe der elliptischen Kurven, angewendet auf das Diffie-Hellman-Verfahren, ist eine hervorragende Allternative zu anderen populären Verfahren gegeben. Bei gleicher Sicherheit wird ein viel geringeres Maß an Speicherplatz benötigt.

Inhaltsverzeichnis

1.1 Einleitung

Bis zum Ende des zweiten Weltkrieges war Verschlüsselung dem Militär und anderen Regierungsinstitutionen weitestgehend vorbehalten. Die älteste überlieferte Verschlüsselungsmethode ist der Caesar Code. Verwendet vom römischen Feldherren Gaius Julius Caesar[1] ging es darum, die einzelnen Buchstaben eines Klartextes (der Text, der verschlüsselt werden soll) jeweils um drei Buchstaben innerhalb des verwendeten Alphabetes zu verschieben und somit einen Geheimtext zu erhalten. Zum Entschlüsseln musste man das Verfahren nur umdrehen und kam auf diese Weise wieder auf den Klartext. Diese Methode nutzte Caesar wenn er mit seinen Offizieren Nachrichten austauschen wollte. Im Laufe der Jahrhunderte verlor dieses Verfahren immer mehr an Sicherheit und wurde durch neue effektivere und schwerer zu entschlüsselnde Verfahren ersetzt. Im zweiten Weltkrieg gipfelte die Kryptographie in einer Maschine namens Enigma auf Seiten der deutschen Wehrmacht.
Im Laufe der folgenden Jahre hat sich jedoch ein immer größerer Bedarf an guten Verschlüsselungsmethoden abgezeichnet. Heutzutage benutzt jeder Mensch, der im Internet surft, mit Kredit- und/oder EC-Karten Geldgeschäfte tätigt, mit dem Handy telefoniert und somit jeder der aktiv am Technologiezeitalter teilnimmt, direkt oder indirekt die aktuellen Verfahren zur Ver- und Entschlüsselung. Weiterhin hat sich durch die rasante Entwicklung der Computerleistung der Anspruch an moderne Verschlüsselungsmethoden stark erhöht. Heutige Verschlüsselungsverfahren lassen sich vorwiegend in zwei Kategorien unterteilen. Bei **symmetrischen Verfahren** geht es darum, dass zwei Kommunikationspartner einen gemeinsamen, geheimen Schlüssel(symmetrisch) verwenden. Die Sicherheit dieser Verfahren ist also im Allgemeinen davon abhängig, wie sicher der Schlüssel ist, wie sicher dieser Schlüssel zu allen Kommunikationspartnern gelangt ist und wie sicher ihn die Partner aufbewahren. Die populärsten symmetrischen Verfahren sind, das DES-Verfahren und das AES-Verfahren.
Die nächste Kategorie sind die **asymmetrischen Verfahren**. Diese sind für die vorliegende Arbeit von größerer Bedeutung. Bei asymmetrischer Verschlüsselung geht es vorwiegend darum, dass jeder Kommunikationsteilnehmer einen persönlichen geheimen Schlüssel besitzt und einen dazu passenden öffentlichen Schlüssel zur Verfügung stellt. Somit besteht bei dieser Methode nicht mehr das Problem der sicheren übermittlung eines Schlüssels. Populäre Methoden die dieses Verfahren verwenden sind z.B. die ElGamal Verschlüsselungsmethode, das RSA Verfahren oder die Diffie - Hellman - Verschlüsselungsmethode. Diffie - Hellman werden wir im 4. Kapitel noch näher erörtern.
Wenn man heutzutage etwas verschlüsselt stellt sich schnell die Frage, wovon die Sicherheit des verwendeten Verfahrens denn nun explizit abhängig ist.[2] An dieser Stelle kommt die Mathematik ins Spiel. Genauer gesagt die Theorie der Einwegfunktionen. Eine Einwegfunktion ist eine Funktion die im Sinne der Komplexitätstheorie(siehe Kapitel 1.2.) "schwer" umzukehren ist. Zur Veranschaulichung stelle man sich ein Telefonbuch vor. Zu einem bekannten Namen die Telefonnummer zu finden ist relativ einfach, aber zu einer bekannten Telefonnummer den Namen zu finden ist äußerst aufwendig. Bei kryptographischen Verfahren geht es grundsätzlich darum, Methoden zu finden, bei der die eine

[1] Er lebte von 100 v.Chr. bis 44 v.Chr.

[2] Wir gehen im Folgenden ausschließlich von asymmetrischen Verfahren aus, so lange nichts anderes explizit erwähnt wird.

Richtung einfach zu berechnen ist und die andere schwierig und ohne eine gewisse Zusatzinformation(den öffentlichen Schlüssel) sogar annähernd unmöglich. In der Mathematik nennt man solche Funktionen, die einen Sonderfall der Einwegfunktionen darstellen, Falltürfunktionen (engl. trapdoor one-way function). Als Beispiel sei an dieser Stelle die Primfaktorenzerlegung genannt. Dies ist das zentrale Problem, auf dem das RSA Verfahren aufbaut. Der eine Weg, also das Produkt von zwei Primzahlen zu bestimmen ist effektiv lösbar, auch bei sehr großen Primzahlen. Versucht man jedoch eine Zahl in ihre Primfaktoren zu zerlegen kann man sehr schnell auf Probleme stoßen, die selbst mit elektronischer Hilfe nicht mehr in angemessener Zeit zu lösen sind. Die Sicherheit dieser Methode ist also direkt von der Länge der Primfaktoren abhängig. Zur Zeit gilt ein 2048 Bit Code also eine Zahl mit 616 Dezimalstellen als mittelfristig sicher.
Eine weitere Einwegfunktion stellt das sogenannte diskrete Logarithmusproblem dar. Dieses Problem ist die Grundlage für das Diffie-Hellman-Verfahren. Die Arbeit wird aufzeigen, warum das diskrete Logarithmusproblem schwer zu lösen ist und wobei es sich darum eigentlich handelt. Weiterhin ist es möglich, das diskrete Logarithmusproblem auf so genannte elliptische Kurven zu übertragen. Auch dieses Problem werden wir genauer erläutern. Die elliptischen Kurven wurden gerade im vergangenen Jahrhundert ausführlich erforscht. 1985 gelang es den Wissenschaftlern Neal Koblitz von der Washingtoner Universität und Victor Miller, der damals für IBM gearbeitet hat, unabhängig voneinander die elliptischen Kurven für die Kryptographie zu entdecken. Anfangs noch nicht überzeugt vom Erfolg ihrer eigenen Idee stellt Elliptic Curve Cryptography(ECC)heute ein zentrales Forschungsgebiet innerhalb der Kryptographie dar. Mit Hilfe der elliptischen Kurven ist es verhältnismäßig leicht, eine Gruppe zu erzeugen, mit der es dann möglich ist, das Diffie-Hellman-Verfahren, oder auch das ElGamal-Protokoll anzuwenden.
Kryptographie mit Hilfe elliptischer Kurven hat einen sehr aktuellen Bezug. So genannte Smartcards überschwemmen förmlich unser tägliches Leben. Kreditkarten, Geldkarten, Krankenversicherungskarten und diverse Zutrittsberechtigungskarten sind nur einige Beispiele. Wären die Daten auf diesen Karten nicht verschlüsselt, wäre es doch zu verlockend für Hobbybetrüger sich dieser Daten mit einem einfachen Chipkartenlesegerät für den PC zu bedienen. Jedoch steigt die Leistung der Computer rasant an und damit müssen die Verschlüsselungsverfahren auf diesen Karten sicherer gemacht werden. Nur gerade bei Smartcards ist der Speicherplatz durch die Bauweise beschränkt. Es ist somit nicht möglich die Schlüssellängen beliebig lang werden zu lassen. An dieser Stelle haben die neuen Verschlüsselungsverfahren auf der Basis elliptischer Kurven einen entscheidenden Vorteil. Um gleiche Sicherheit zu garantieren, benötigen sie kürzere Schlüssellängen, als andere Verfahren, die z.B. auf dem Problem der Primfaktorenzerlegung beruhen.

1.2 Komplexitätstheorie

Die Komplexitätstheorie ist eine Disziplin, die zwischen der Informatik und Mathematik aufzuhängen ist. Dieses Kapitel soll jedoch nur erläutern, was Komplexitätstheorie ist und womit sich selbige befasst, um ein besseres Verständnis vor allem für die Problematik des diskreten Logarithmusproblem zu liefern. Dabei wird nicht in besonderer Tiefe auf die Mathematik eingegangen. Es soll stattdessen der intuitive Zugriff auf Berechenbarkeitsprobleme in die richtigen Bahnen gelenkt werden.
Um den Einstieg zu erleichtern soll folgendes Beispiel dienen. Einen Baum zu fällen erfordert Kraft, die von dem benutzten Hilfsmittel abhängig ist. Mit einer Kettensäge geht es schneller als mit einer Axt und es erfordert weniger Kraft. Hat man jedoch nur eine Pfeile zur Verfügung ist das Problem des Baumfällens nicht mehr effizient lösbar. Man kann also sagen, dass bei dem Problem des Baumfällens die benötigten Ressourcen[3] und die benötigte Zeit bis der Baum fällt, direkt vom gewählten Hilfsmittel abhängig sind. Geht es jedoch darum Holz in Scheite zu spalten, ist eine Axt offensichtlich besser geeignet als eine Kettensäge. Dieses Beispiel verdeutlicht, dass auch die Güte der Hilfsmittel direkt mit dem Problem zusammenhängt.
Generell geht es in der Komplexitätstheorie ebenfalls um die Frage nach benötigten Ressourcen und benötigter Zeit. Man betrachtet das mathematische Modell eines Problems und dann sucht man nach Algorithmen um das Problem zu lösen. Jeder Algorithmus ist dann nach benötigter Zeit und benötigten Ressourcen[4] einzuteilen. Aus diesem Grund ist es offensichtlich, dass endgültige Aussagen über die Berechnungskomplexität oder auch die "Härte" eines Problems schwer zu treffen sind. Es ist nämlich durchaus möglich, das für ein bestehendes Problem im Laufe der Zeit ein effektiverer, oder falls es für das Problem noch gar keinen gibt überhaupt einen, Algorithmus gefunden wird. Um die Effizienz eines Algorithmus nach Zeit und Ressourcen festzustellen, bedient man sich gerne dem theoretischen Konstrukt der Turingmaschine. Eine Turingmaschine ist ein simples und abstraktes Computermodell. Bei Befehlen für eine Turingmaschine kann man im Gegensatz zu einer konkreten Maschine einen Einheitswert ansetzen.[5] Dies erleichtert sehr stark das Finden einer allgemeinen Aussage. Kurz sollen an dieser Stelle die technischen Details einer Turingmaschine beschrieben werden, da sie wirklich eine zentrale Rolle als Hilfsmittel auf dem Gebiet der Komplexitätstheorie darstellt. Eine Turingmaschine ist mit wenigstens einem Arbeitsband ausgestattet, das beidseitig unendlich lang und in Bandfelder unterteilt ist. Dieses Band hat einen Lese- und einen Schreib- Kopf, der stets auf genau einem Feld dieses Bandes steht. Jedes Feld kann ein Symbol enthalten. Die Abwesenheit eines Symbols in einem Feld wird durch ein spezielles Symbol angezeigt, dem Leerzeichen □. Dieses Zeichen kann die Maschine zwar verarbeiten, es ist aber nicht vorgesehen, dieses Symbol einzugeben. Nun ist es mit diesem Modell möglich, eine Berechnung der Schritte vorzunehmen, die nötig sind um einen Algorithmus zu lösen. Fiktiv wird ein Eingabeband in die Turingmaschine eingeführt, gleichzeitig sind alle anderen Bandfelder leer. Die Berechnung erfolgt nun über das Lese- Schreib- Band und am Ende der Berechnung hält die Maschine und es kommt die Ausgabe auf dem Ausgabeband heraus. Viele technische Variationen sind möglich auf die an dieser Stelle jedoch nicht näher eingegangen werden

[3] Die Ressourcen sind in diesem Fall die eigen aufzubringende Kraft.
[4] Hier ist mit Ressourcen generell die Rechenleistung gemeint.
[5] Auch genannt uniformes Kostenmaß.

soll.
Die Berechnung innerhalb einer Turingmaschine soll noch etwas genauer erörtert werden. Jeder Berechnungsschritt der Turingmaschine besteht darin, dass jeder Kopf das Symbol im gerade besuchten Feld liest und es dann entweder überschreibt oder unverändert lässt. Danach wandert der Kopf entweder nach rechts, nach links, oder bleibt an Ort und Stelle. Gleichzeitig wird nun die Maschine ihren aktuellen, inneren Zustand, der in ihrem inneren Gedächtnis (Vorgang in engl. "finite ontrol") gespeichert ist ändern oder beibehalten.[6]
Nun ist es also möglich, die benötigten Ressourcen[7] und die benötigte Zeit[8], die ein Algorithmus braucht um ein Problem zu lösen, zu ermitteln. Zwei weitere Punkte spielen bei der Bestimmung der Härte eine entscheidende Rolle. Zum einen das Berechnungsmodell oder das algorithmische Gerät, in unserem Fall die Turingmaschine. Zum anderen das Berechnungsparadigma des Berechnungsmodells. Das Berechnungsparadigma wird bei der Turingmaschine durch die angesprochenen verschiedenen Varianten verkörpert, z.B. gibt es deterministische oder nichtdeterministische Turingmaschinen.
Weiterhin gibt es die Möglichkeit, die Komplexität eines Problems in verschiedene Komplexitätsklassen einzuteilen. Dies ist sinnvoll, da es somit möglich wird, die Härte verschiedener Probleme zu vergleichen.

1.2.1 Komplexitätsklassen

Die Komplexität eines Problems lässt sich mit Hilfe von Komplexitätsklassen beschreiben oder klassifizieren. Für eine Komplexitätsklasse Ψ und ein Problem A erfordert der nötige Beweis um A in Ψ einzuordnen Folgendes. Ψ muss gleichzeitig obere und untere Schranke für A liefern. Das bedeutet, dass man beweisen muss, dass mindestens ein Algorithmus A löst, und dabei höchstens so viele Komplexitätsressourcen verbraucht, wie Ψ zur Verfügung stellt. Weiterhin muss man beweisen, dass alle Algorithmen, die A lösen wenigstens so viele Komplexitätsressourcen verbrauchen, wie Ψ zur Verfügung stellt. Dieses Beweisstück ist üblicherweise viel anspruchsvoller, als das Beweisen der oberen Schranke. Zusammenfassend muss man also argumentieren, um A Ψ zuordnen zu können, dass es keinen Algorithmus gibt, der A lösen kann und gleichzeitig weniger Ressourcen braucht, als Ψ zur Verfügung stellt. Daraus folgt, dass sich ein Problem B aus einer Komplexitätsklasse Θ von geringerer Härte als Problem A ebenfalls Ψ zuordnen lässt. Der umgekehrte Fall gilt jedoch nicht automatisch. Hier ist es davon abhängig, ob B alle Ressourcen benötigt die Θ zur Verfügung stellt, und somit die obere Schranke von B gleich der unteren Schranke von Θ ist. Ist das der Fall, ist B gleichzeitig ein Problem, dass Θ als Komplexitätsklasse in seiner Ganzheit verkörpert. Weiterhin ist es dann unmöglich, dass A Θ zuzuordnen ist.
Einige Komplexitätsklassen seien hier vorgestellt. Die Komplexitätsklasse P ist gemeinhin als die Klasse definiert, welche die Probleme enthält, die in Polynomialzeit für deterministische Turingmaschinen lösbar sind.[9] Diese Komplexitätsklasse enthält im allgemeinen

[6] Dem interessierten Leser sei das Kapitel über Turingmaschinen in [12] nahegelegt

[7] Bei der Turingmaschine wäre das Maß für die Ressourcen, die Anzahl der verwendeten Bandzellen, die die Turingmaschine nutzt um den Algorithmus zu lösen.

[8] Bei der Turingmaschine wäre das Maß für die Zeit, die Anzahl der Schritte, die die Turingmaschine braucht um den Algorithmus zu lösen.

[9] P wird in der Literatur auch mit PTIME bezeichnet

alle Probleme, die als "praktisch lösbar" gelten. Eine weitere Komplexitätsklasse der Probleme die von einer nichtdeterministischen Turingmaschine in Polynomialzeit entschieden werden können wird gemeinhin als NP definiert. NP-vollständige Probleme lassen sich **vermutlich** nicht effizient lösen.[10] Vermutlich deswegen, weil allgemein angenommen wird, aber nicht bewiesen ist, dass es keine effizienteren Algorithmen gibt, die das diskrete Logarithmusproblem lösen, als diejenigen die sich nur in Exponentialzeit lösen lassen.

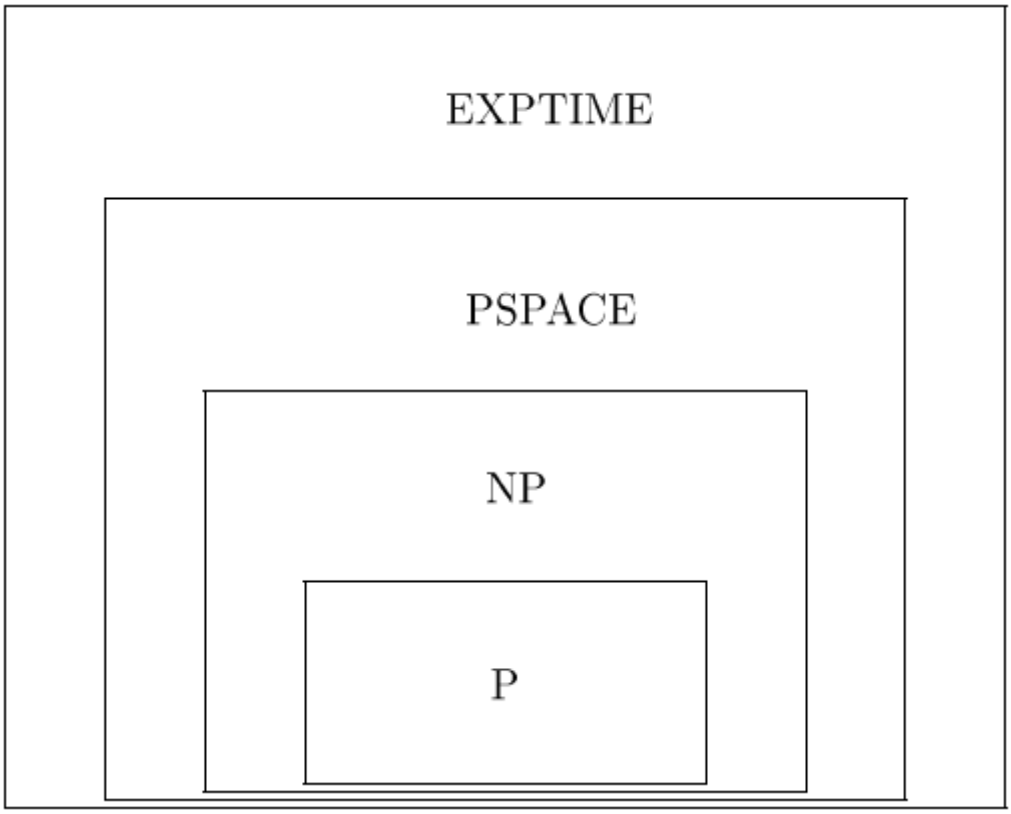


Abbildung 1.1: Hier kann man erkennen, dass die Komplexitätsklassen von geringer Härte sich in Komplexitätsklassen größerer Härte überführen lassen. P in NP in PSPACE in EXPTIME.

Auf die polynomiale und exponentielle Laufzeit wollen wir am Ende von Kapitel 1.2.3 noch genauer mit einigen Beispielen eingehen um dem Leser einen besseren Zugang zu ermöglichen.

1.2.2 Algorithmen

Nachdem wir jetzt den Begriff Algorithmus häufig verwendet haben soll eine Erläuterung dieses Begriffes dem Leser nicht vorenthalten werden. Ein Algorithmus ist eine endliche Folge von Prozeduren oder Regeln, die für ein Problem formuliert dieses Problem lösen soll.

[10] Vollständig meint in diesem Fall, die Probleme die alle Ressourcen benötigen, die NP zur Verfügung stellt.

Eine wichtige Aufgabe der Forschung ist es also genau solche Formulierungen aufzustellen. Einer der bekanntesten Algorithmen ist der Euklidische Algorithmus, der das Problem des größten gemeinsamen Teilers zweier gegebener ganzen Zahlen löst. Er taucht das erste Mal im Buch der Elemente von Euklid von Alexandria[11] auf. Auch heute findet er vielerorts noch regen Gebrauch.

```
EUKLID(n,m){
if(m=0) return n;
else return EUKLID(m,n mod m);
}
```

Im nächsten Kapitel werden wir uns etwas ausführlicher mit einem anderen Algorithmus beschäftigen und zwar dem "Baby-Step Giant-Step" Algorithmus, der eine Möglichkeit zur Berechnung des diskreten Logarithmus eines Elementes einer zyklischen Gruppe darstellt. In Kapitel 1.4 werden wir sehen, warum gerade dieser Algorithmus für unsere Arbeit eine zentrale Rolle spielt.

1.2.3 $\mathcal{O}$-Notation

Die $\mathcal{O}$-Notation, auch Landau Notation genannt, wird in der Mathematik verwendet um Aussagen über Laufzeiten von Funktionen zu machen. Ist $g : \mathbb{N} \to \mathbb{N}$ eine gegebene Funktion, so ist die Komplexitätsklasse $\mathcal{O}(g)$ definiert als

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{N} | (\exists c > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[f(n) \leq c \cdot g(n)]\}.$$

Verbal bedeutet diese Definition, dass eine Funktion $f \in \mathcal{O}(g)$ asymptotisch nicht schneller wächst als g. Somit ist es möglich über die Landau Notation in Kurzform zu sagen, in welcher Klasse die zu betrachtende Funktion liegt. Dabei werden additive Konstanten, konstante Faktoren und endlich viele Ausnahmen vernachlässigt.
Ein paar Beispiele sollen das verdeutlichen.

- $f \in \mathcal{O}(log\ n)$ bedeutet für f ein logarithmisches Wachstum.
- $f \in \mathcal{O}(n)$ bedeutet für f ein lineares Wachstum.
- $f \in \mathcal{O}(n^2)$ bedeutet für f ein quadratisches Wachstum.
- $f \in \mathcal{O}(2^n)$ bedeutet für f ein exponentielles Wachstum.

[11] etwa 325 bis 265 v.Chr.

Nachdem nun die Landau Notation bekannt ist wollen wir noch einmal zu den Komplexitätsklassen zurückkehren. Ein erstes Beispiel sei die Funktion

$$g(n) \;=\; 2n^2 + 7n - 10$$

und

$$f(n) \;=\; n^2,$$

es gilt $g(n) \;\in\; \mathcal{O}(f(n))$, da mit $c \;=\; 3$ und $\forall\; n \;\geq\; 5$

$$2n^2 \;+\; 7n \;-\; 10 \;\leq\; c \cdot n^2$$

gilt. Somit kann man sagen, dass die Funktion $g(n)$ in $\mathcal{O}(n^2)$ liegt. Abbildung 1.2 soll nun aufzeigen, wie sich verschiedene Komplexitätsklassen auf einer theoretischen Maschine bezüglich ihrer jeweiligen Laufzeit und der jeweiligen Eingabe verhalten.[12] In diesem Fall sei die theoretische Maschine ein Computer, der in der Lage ist, eine Million Anweisungen pro Sekunde zu verarbeiten. Es wird schnell deutlich, dass die Komplexitätsklassen, die eine exponentielle Laufzeit haben zeitlich gesehen denen mit polynomialer Laufzeit unterlegen sind. Selbst bei unseren, im Vergleich zur Realität, sehr klein gewählten n sind die Laufzeiten schon inakzeptabel. In der Tabelle wird durch verschiedene $t(n)$ abhängig von gegebenen n die Zeit aufgetragen, die der Computer braucht um im schlechtesten Fall(engl. worst case), also mit der größtmöglichen Anzahl an Schritten, zu einem Ergebnis zu kommen.
Weiterhin ist auch hinsichtlich der rasanten Entwicklung moderner Computer keineswegs damit zu rechnen, dass die benötigte Zeit zur Berechnung exponentieller Laufzeitalgorithmen ebenso rasant abnimmt. Das liegt daran, dass die Leistung nicht direkt als Faktor zur Größe der Probleminstanz hinzugefügt werden darf. Bei den exponentiellen Problemen ist es sogar so, dass bei 1000-facher Erhöhung der Rechenleistung lediglich 10 zur Größe einer Probleminstanz, die ein 2^n-zeitbeschränkter Algorithmus innerhalb einer Stunde lösen kann, addiert wird. Bei einem 3^n-zeitbeschränktem Algorithmus ist der Summand sogar lediglich ≈ 6.

Nun soll noch ein Beispiel zum besseren Verständnis folgen. Wir betrachten die Multiplikation zweier Zahlen mit der Länge n-Bit. Es erfordert n Multiplikationen das Ergebnis zu berechnen, da jede Ziffer der einen Zahl mit jeder Ziffer der anderen Zahl multipliziert werden muss. Somit ist die Multiplikation zweier Zahlen ein Beispiel für die Komplexitätsklasse $\mathcal{O}(n)$.

[12] Die Tabelle ist übernommen aus [12] und die größe der Eingabe wird beschrieben durch n.

t(n)	n=10	n=20	n=30	n=40	n=50	n=60
n	0.00001 Sek	0.00002 Sek	0.00003 Sek	0.00004 Sek	0.00005 Sek	0.00006 Sek
n^2	0.0001 Sek	0.0004 Sek	0.0009 Sek	0.0016 Sek	0.0025 Sek	0.0036 Sek
n^3	0.001 Sek	0.008 Sek	0.027 Sek	0.064 Sek	0.125 Sek	0.256 Sek
n^5	0.1 Sek	3.2 Sek	24.3 Sek	1.7 Min	5.2 Min	13.0 Min
2^n	0.001 Sek	1.0 Sek	17.9 Min	12.7 Tage	35.7 Jahre	366 Jhdte.
3^n	0.059 Sek	58 Min	6.5 Jahre	3855 Jhdte.	$2 \cdot 10^8$ Jhdte.	$1.3 \cdot 10^{13}$ Jhdte.

Abbildung 1.2: Theoretische Laufzeiten von Algorithmen/Funktionen verschiedener Komplexitätsklassen.

1.3 Das diskrete Logarithmusproblem

Das Problem des diskreten Logarithmus wird im Allgemeinen für viele Verfahren der Kryptographie verwendet und im Besonderen auch für das Diffie - Hellman - Protokoll. Zunächst wollen wir einige mathematische Grundlagen einführen.

Definition 1.3.1 Gruppe:
Ein Paar $(G, \otimes)$ aus einer Menge G und einer inneren zweistelligen Verknüpfung $\otimes : G \otimes G \to G : (a, b) \mapsto a \otimes b$ heißt **Gruppe**, wenn folgende Axiome erfüllt sind:

- Assoziativität: für alle Gruppenelemente a, b und c gilt: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$.
- Neutrales Element: Es gibt ein neutrales Element $\mathbb{O}$ in G, so dass für alle Gruppenelemente a gilt: $a \otimes \mathbb{O} = \mathbb{O} \otimes a = a$.
- Inverses Element: Zu jedem Gruppenelement a existiert ein Element a^{-1} mit $a \otimes a^{-1} = a^{-1} \otimes a = \mathbb{O}$.

Eine Gruppe $(G, \otimes)$ heißt kommutativ oder abelsch, wenn die Verknüpfung $\otimes$ symmetrisch ist, also wenn folgendes Axiom ebenfalls gilt:

- Kommutativität: Für alle Gruppenelemente a und b gilt $a \otimes b = b \otimes a$

Wichtig für diese Definition ist, dass die Verknüpfung $\otimes$ eine innere Verknüpfung sein muss. Das bedeutet, dass $\otimes\, a, b \in G$ wieder auf ein Element abbildet, welches in G liegt. Es muss gelten $a \otimes b \in G$.

Im Folgenden sei $\mathbb{Z}$ die Menge aller ganzen Zahlen.

Definition 1.3.2 Ringe und Körper:
Ein Tripel $(R, +, \cdot)$ heißt Ring, wenn gilt:

- $(R, +)$ ist eine abelsche Gruppe.
- Die Multiplikation ist assoziativ.
- Es existiert ein neutrales Element $\mathbb{O}$ bezüglich der Multiplikation $a \cdot \mathbb{O} = \mathbb{O} \cdot a = a$
- Distributivgesetz:

$$(a + b) \cdot c = (a \cdot c) + (b \cdot c);\ a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

Definition 1.3.3 Körper:
Ein Körper K ist ein kommutativer Ring, so dass jedes Element von $k \neq 0$ invertierbar ist.

Definition 1.3.4 endliche Gruppen, Untergruppen:
Besteht die Gruppe G nur aus endlich vielen Elementen, dann ist die Ordnung $ord(G)$ gleich der Anzahl der Elemente der Gruppe. Zum Beispiel ist $ord(\mathbb{Z}_n) = n$. Weiterhin heißt eine Teilmenge H von G Untergruppe von G, falls H eine Gruppe bezüglich der auf H eingeschränkten Gruppenoperationen von G ist.

Definition 1.3.5 Erzeuger und zyklische Gruppe:
Existiert ein $x \in G$ und es gilt $< x > = G$, dann nennt man G **zyklisch**. Existiert so ein x, dann nennt man x Erzeuger von G oder auch G wird von x erzeugt.

Folgende Beispiele sollen zum besseren Verständnis dienen:

- Sei $ka = a + \ldots + a$ (k mal), dann ist die Teilmenge aller Vielfachen von a $< a > = \{ka : k \in \mathbb{Z}\}$ eine Untergruppe von G erzeugt durch a
- Sei $kb = b \cdot \ldots \cdot b$ (k mal) dann ist die Menge $< b > = \{b^k : k \in \mathbb{Z}\}$ eine Untergruppe von G erzeugt durch b

- Sei p eine Primzahl, dann ist K ein Körper mit $\mathrm{ord}(K_p) = p$, es gilt:

$$p \cdot y \ = \ 0 \ mod \ p \ \forall y \ \in K$$

$$a^p \ = \ a \ mod \ p \ \forall a \ \in K_p$$

Definition 1.3.6 Kongruenz:
Zwei ganze Zahlen a und b sind kongruent modulo m, wenn m die Differenz $a \ - \ b$ teilt. Zwei Zahlen a und b sind inkongruent modulo m, wenn m die Differenz $a \ - \ b$ nicht teilt. In mathematischer Notation bedeutet das:

$$a \equiv b \ (mod \ m) \Leftrightarrow m| \ (b \ - \ a) \Leftrightarrow \exists k \in \mathbb{Z} : a = km + b$$

$$a \not\equiv b \ (mod \ m) \Leftrightarrow m \nmid \ (a \ - \ b) \Leftrightarrow \forall k \in \mathbb{Z} : a \neq km + b$$

Definition 1.3.7 Restklassen:
Die Kongruenz ist eine äquivalenzrelation. Sie hat somit folgende Eigenschaften:

- Reflexivität: $a \equiv a \ (mod \ m) \ \forall \ a \in \mathbb{Z}$
- Symmetrie: $a \equiv b \ (mod \ m) \Rightarrow b \equiv a \ (mod \ m) \ \forall \ (a, b) \in \mathbb{Z}^2$
- Transitivität: $a \equiv b \ (mod \ m) \wedge b \equiv c \ (mod \ m) \Rightarrow a \equiv c \ (mod \ m) \ \forall \ (a, b, c) \in \mathbb{Z}^3$

Sei nun $m \geq 1$ und $m \in \mathbb{N} \ \wedge \ a \in \mathbb{Z}$. Die **Restklasse** von $a \ mod \ m$ (oft bezeichnet mit $\overline{a}$) ist die Menge aller ganzen Zahlen k für die gilt $a \equiv k \ mod \ m$. Die Menge aller **Restklassen** *modulo* m wird oft mit $\mathbb{Z}_{/m\mathbb{Z}}$ bezeichnet und heißt Restklassenring.

Mit diesem Wissen ausgestattet wenden wir uns nun dem diskreten Logarithmusproblem zu. Sei p eine Primzahl und G eine abelsche zyklische Gruppe der Ordnung n. Sei weiter a der Erzeuger dieser Gruppe und x eine ganze Zahl. Die Funktion $exp : \ G \ \rightarrow \ G \ : \ x \ \mapsto \ a^x \ mod \ p$ heißt diskrete Exponentialfunktion. Die Umkehrfunktion von exp heißt diskrete Logarithmusfunktion. Da a ein erzeugendes Element von G ist, gibt es, da sich jedes Element von G als Potenz von a darstellen lässt, zu jedem Element aus G den diskreten Logarithmus. Die kleinste natürliche Zahl x mit $y \ = \ a^x \ mod \ p$ heißt der diskrete Logarithmus von y zur Basis a.Weiter sei festgehalten, dass man a auch als Primitivwurzel von n bezeichnet. Am Rande sei erwähnt, dass es, sobald a kein Erzeuger von G ist, nicht klar ist, ob der diskrete Logarithmus überhaupt existiert. In der Kryptographie kann man aber im allgemeinen davon ausgehen, das es, sobald es um das diskrete Logarithmusproblem geht, der diskrete Logarithmus auch existiert. Dieses Entscheidungsproblem spielt für uns also keine Rolle. Bis heute ist kein Algorithmus bekannt, der dieses Problem, also das Finden des kleinstmöglichen x, effizient löst. Zur Verdeutlichung sei erwähnt, dass es selbstverständlich möglich ist, durch das Verfahren der Enumeration, also durch das reine Ausprobieren sämtlicher Möglichkeiten auf eine Lösung zu stoßen. Hierbei gilt, da a ein Erzeuger von G ist, hat a die Ordnung $p - 1$, daraus folgt, dass $a^{p-1} \ mod \ p \ = \ a^0 \ = \ 1$

ist. Beim Verfahren der Enumeration muss man also y nur in der Menge $\{0, \cdots, p-2\}$ suchen. Bei dieser Suche sind dann $x-1$ Multiplikationen und x Vergleiche in G nötig. Weiterhin muss man die Elemente y, a und a^x speichern. Bei kryptographischen Verfahren wird mit Zahlen x gearbeitet, die größer als 2^{160} sind. An dieser Stelle wird deutlich, dass das Enumerationsverfahren hier nicht effizient ist.
Ein etwas effektiveres Verfahren ist der Babystep - Giantstep Algorithmus. Hierbei sind weniger Operationen nötig als beim Verfahren der Enumeration, jedoch hat dieser Algorithmus einen größeren Speicherplatzbedarf. Sei $m = \sqrt{n}$. Nun nehmen wir den Ansatz $x = qm + r,\ 0 \leq r < m$. Hierbei ist q also der Quotient und r der Rest der Division von x durch m. Mit Hilfe unseres Algorithmus wollen wir nun q und r berechnen und zwar auf folgende Art und Weise. Es gilt

$$a^{qm+r} = a^x = y.$$

Hieraus folgt

$$(a^m)q = y \cdot a^{-r}.$$

Nun wird zunächst die Menge der Babysteps berechnet

$$B = \{(y \cdot a^{-r}, r) : 0 \leq r < m\}.$$

Findet man auf diese Weise ein Paar $(1, r)$, so kann man $x = r$ setzen und hat das Diskrete Logarithmusproblem gelöst. Hat man dieses Paar an dieser Stelle nicht gefunden, geht es weiter. Man bestimmt

$$\delta = a^m,$$

dann wird geprüft, ob für $q = 1, 2, 3, \ldots$ das Gruppenelement δ^q als erste Komponente eines Elements von B vorkommt, ob also ein Paar (δ^q, r) zu B gehört.[13] Sobald man ein solches Paar gefunden hat, gilt

$$y \cdot a^{-r} = \delta^q = a^{q \cdot m}.$$

Somit hat man also den diskreten Logarithmus $x = q \cdot m + r$ gefunden.[14]

1.4 Asymmetrische Verschlüsselung als Beispiel Diffie - Hellmann - Verfahren

Lange Zeit gab es in der Kryptographie ein ganz entscheidendes Problem. Bei der symmetrischen Verschlüsselung muss der geheime Schlüssel zwischen zwei Kommunikationspartnern ausgetauscht werden, **bevor** sie damit beginnen, Nachrichten untereinander auszutauschen. Die Kommunikationspartner seien im Folgenden Alice und Bob getauft. Wie schaffen Alice und Bob es nun besagten Schlüssel auszutauschen ohne sich persönlich zu treffen? Dies muss Mindestanforderung an ein kryptographisches Verfahren sein. Man

[13] Die Berechnung von δ^q und $q = 1, 2, 3, \ldots$ bezeichnet man als Giantsteps.
[14] Dieser Algorithmus wurde übernommen aus [3].

stelle sich einmal vor, dass ein General eine Nachricht an jeden Einzelnen seiner Untergebenen Soldaten senden will, die selbstverständlich geheim ist und somit verschlüsselt sein soll. Das bedeutet, dass falls der General zur Verschlüsselung seiner Nachricht ein symmetrisches Kryptoverfahren benutzen würde, jeder Empfänger der Nachricht ebenfalls den geheimen Schlüssel zum entschlüsseln der Nachricht haben müsste. Gehen wir einmal davon aus, dass jeder Untergebene diesen Schlüssel besitzt[15], dann wäre ein weiteres Problem die Geheimhaltung des Schlüssels. Angenommen er wäre bei einer Zahl von 50.000 Untergebenen ebenso oft verteilt worden, wer kann da noch garantieren, dass ein feindlicher Geheimdienst den Schlüssel nicht schon längst in seinen Besitz bringen konnte. Somit wäre die Sicherheit des Verfahrens nicht mehr gegeben. Diese paradoxe Situation ist als das Problem des Schlüsseltausches (englisch "secret-key agreement problem") bekannt. Schon seit den Anfängen der Kryptographie gilt dieses Problem als unlösbar. Versuche trotz dieses Problems zu Verschlüsseln bedeutete gerade im Zuge des II. Weltkrieges auf deutscher Seite ewiges Wälzen und Benutzen von Codebüchern zusammen mit der Enigma. Sobald ein Codebuch verschwand, war die Sicherheit selbstverständlich nicht mehr gegeben.
1976 gab es eine Veröffentlichung von Whitfield Diffie und Martin Hellmann die eine Revolution der Kryptographie darstellen sollte. Interessanterweise wurde im Jahre 1997 bekannt, dass der Auftrag zur Entwicklung einer neuen Kryptomethode vom britischen Government Communications Headquarters[16] kam. Bei dem von Diffie und Hellmann entwickelten Verfahren geht es darum, dass Alice und Bob den Kryptoschlüssel nicht von vornherein festlegen und austauschen müssen, sondern ihn gemeinsam entwickeln. Hierbei können alle zur Entwicklung des Schlüssels nötigen Nachrichten, die Alice und Bob austauschen müssen, bis auf das letzte Bit von einem potentiellen Angreifer[17] abgefangen werden, ohne dass Eva damit auch nur das Geringste anfangen kann. Dies beruht auf der im Allgemeinen angenommenen Unlösbarkeit des diskreten Logarithmusproblems. Wie funktioniert dieses Verfahren nun im Detail?

1. Zunächst einigen sich Alice und Bob auf eine Primzahl p und eine Primitivwurzel $g \; mod \; p \; mit \; 2 \leq g \leq p-2$. Diese Parameter sind also öffentlich zugänglich, da sie über einen öffentlichen Kanal ausgetauscht werden.

2. Nun müssen Alice und Bob sich je eine eigene Zufallszahl erzeugen. Diese Zufallszahl ist jeweils für Bob und Alice der private Schlüssel und darf nicht öffentlich zugänglich gemacht werden. Bob muss zu keinem Zeitpunkt explizite Informationen über Alice privaten Schlüssel erhalten und umgekehrt. Alice privater Schlüssel sei im Folgenden a und Bobs privater Schlüssel im Folgenden b genannt. Weiter sollen a und b aus $\{1, \ldots, p-2\}$ gewählt werden.

3. Nun berechnet Alice ihren öffentlichen Schlüssel A:

$$A = g^a \; mod \; p$$

[15]Das für sich würde schon einen riesigen Aufwand bedeuten und falls der Schlüssel ebenfalls über eine Nachricht verteilt worden wäre, welchen Schlüssel sollte man zur Verschlüsselung dieser Nachricht verwenden?

[16]Das ist eine britische Regierungsbehörde mit den Fachgebieten Nachrichtendienst und Sicherheitsdienst.

[17]Im Folgenden wollen wir diesen Potentiellen Angreifer Eva nennen.

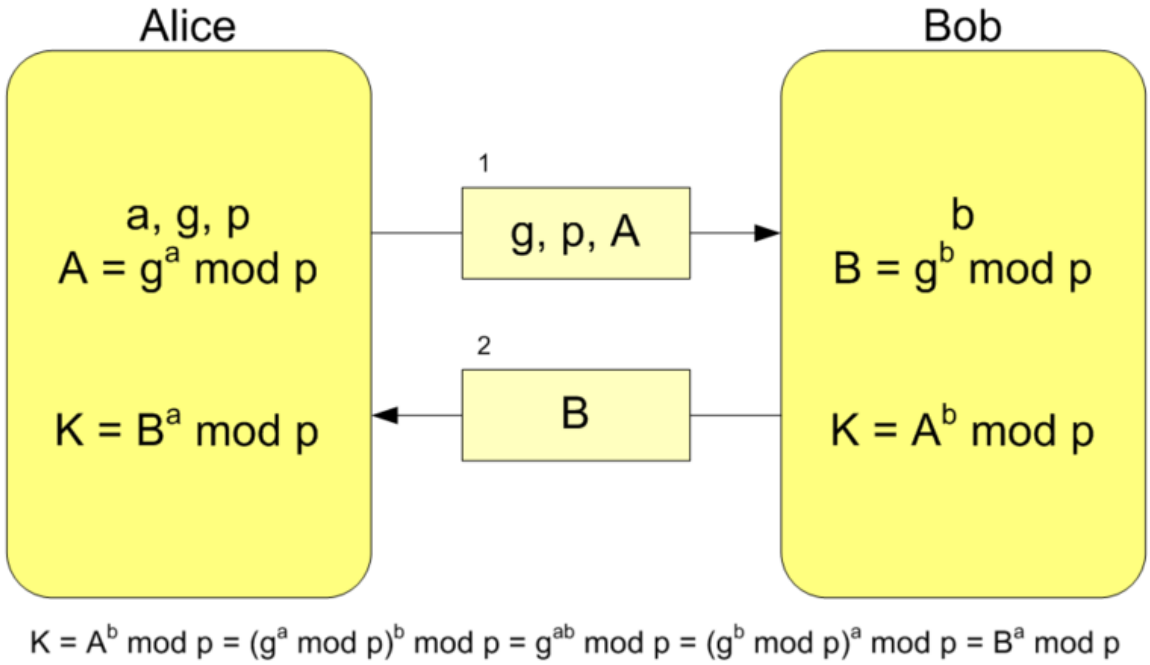


Abbildung 1.3: Das Verfahren des Diffie - Hellman - Schlüsseltausches

Bob berechnet seinen öffentlichen Schlüssel B:

$$B \ = \ g^b \ mod \ p$$

Jetzt sendet Alice A an Bob und Bob sendet B an Allice.[18]

4. Nun wird der gemeinsame geheime Schlüssel K berechnet, für Alice heißt das:

$$K \ = \ B^a \ mod \ p$$

und für Bob:

$$K \ = \ A^b \ mod \ p$$

Alice und Bob haben nun den selben Schlüssel K berechnet, da

$$K \ = \ B^a \ mod \ p \ = \ (g^b \ mod \ p)^a \ mod \ p \ = \ g^{a \cdot b} \ mod \ p$$

und

$$K \ = \ A^b \ mod \ p \ = \ (g^a \ mod \ p)^b \ mod \ p \ = \ g^{a \cdot b} \mod \ p.$$

Die Berechnung von K ist mit dem effizienten Algorithmus Square and Multiply einfach zu berechnen.[19] Wir erinnern uns, dass Eva zu diesem Zeitpunkt im Besitz sein kann von

[18] Da die Möglichkeit der Verschlüsselung zu diesem Zeitpunkt noch nicht gegeben ist, sind folglich auch A und B öffentlich zugänglich.

[19] Dieser Algorithmus ist ein sehr effizientes Verfahren zur Berechnung hoher Potenzen natürlicher Zahlen. Siehe hierzu auch [13].

g, p, A *und* B. Um a und b zu bestimmen und somit den geheimen Schlüssel K selber berechnen zu können muss Eva also in der Lage sein, folgende Gleichungen zu lösen.

$$a = log_g \cdot A \; mod \; p$$
$$b = log_g \cdot B \; mod \; p$$

Man sieht also, das zum aggressiven Entschlüsseln des Diffie - Hellman - Protokolls das Lösen des diskreten Logarithmusproblems erforderlich ist. Weiter ist es aber wichtig zu erwähnen, dass bis heute nicht bewiesen werden konnte, ob es nicht vielleicht andere Möglichkeiten gibt das Diffie - Hellman - Problem zu lösen. Es wird jedoch gemeinhin angenommen, dass dem so ist. Darüberhinaus ist es ebenfalls nocht nicht bewiesen, ob das Problem des diskreten Logarithmus wirklich ein "Hartes" Problem darstellt.[20] Jedoch wird zum heutigen Zeitpunkt auch dies gemeinhin angenommen.

1.4.1 Asymmetrische Kryptographie

Nachdem Diffie und Hellman ihr Verfahren vorgestellt haben, war der Weg für asymmetrische Verschlüsselungsverfahren geebnet. Das Verfahren funktioniert folgendermaßen. Will Alice Bob eine Nachricht schicken, dann nimmt sie Bobs öffentlichen Schlüssel und berechnet mit diesem Schlüssel den gemeinsamen, geheimen Schlüssel für den Schlüsselaustausch. So geschehen ist Alice nun in der Lage, mit dem gemeinsamen, geheimen Schlüssel den Klartext zu verschlüsseln. Wenn Bob nun den von Alice erhaltenen Geheimtext entschlüsseln will, sucht er einfach ihren öffentlichen Schlüssel und berechnet damit ebenfalls den gemeinsamen, geheimen Schlüssel, mit welchem er dann in der Lage ist, den von Alice erzeugten Geheimtext zu entschlüsseln.[21] Der Vorteil für dieses Verfahren liegt vor allem darin, dass eine vorherige Absprache zum Nachrichtenaustausch nicht mehr nötig ist, da ein geheimer Schlüssel nicht mehr unmittelbar ausgetauscht werden muss. Weiterhin ist es einfach, auch mit neuen Mitgliedern des Kommunikationskreises Nachrichten auszutauschen. Ein neues Mitglied muss, wie schon Alice und Bob, nur seinen eigenen öffentlichen Schlüssel erstellen und diesen öffentlich zugänglich machen. Schon ist es Alice und Bob möglich auch mit diesem neuen Mitglied verschlüsselt zu kommunizieren. Das "secret-key agreement problem" spielt somit keine Rolle mehr und es ist möglich geheim zu kommunizieren. Ein großer Nachteil der asymmetrischen Verschlüsselung sei an dieser Stelle noch erwähnt. Da die Funktionswerte, die es zu berechnen gilt, zwar nach heutigen Standards effektiv zu berechnen sind, die entstehenden Schlüssel aber aufgrund der Sicherheitsanforderungen sehr lang werden können ist es somit stark von der Länge des Klartextes abhängig, wie viel Rechenaufwand zum Ver- und Entschlüsseln nötig ist. Aus diesem Grund ist man auf die Idee der hybriden Verfahren gekommen. Hierbei wird der Schlüssel des symmetrischen Verfahrens mit einer asymmetrischen Methode verschlüsselt und verschickt. Der Klartext kann daraufhin mit der symmetrischen Verschlüsselungsmethode behandelt werden. Zwar ist selbsverständlich auch die benötigte Laufzeit eines symmetrischen Protokolls abhängig von der Länge des Klartextes, gleichzeitig sind die

[20] Siehe hierzu auch das Kapitel 1.3.

[21] Die Vorgehensweise für das Diffie - Hellman - Protokoll wurde oben erläutert.

Symmetrische Schlüssellänge	Asymmetrische Schlüssellänge
56 Bit	384 Bit
64 Bit	512 bit
80 Bit	768 Bit
112 Bit	1792 Bit
128 Bit	2304 Bit

Tabelle 1.1: Vergleich symmetrischer und asymmetrischer Schlüssellängen

benötigten symmetrischen Schlüssel aber auch kürzer bei gleicher gegebener Sicherheit als die asymmetrischen Schlüssel. Tabelle 1.1 stellt beispielhaft einige symmetrische und asymmetrische Schlüssellängen, die jeweils gegenübergestellt für ein gleiches Maß an Sicherheit sorgen, dar. Offensichtlich geht es nun schneller einen Klartext mit fixer Länge z.B. mit einem 128 Bit Schlüssel zu ver- und entschlüsseln, als mit einem 2304 Bit langen Schlüssel. Die Werte der Tabelle sind übernommen aus [21].

1.5 Elliptische Kurven

Elliptische Kurven können uns eine Gruppe liefern, mit deren Hilfe das diskrete Logarithmusproblem gewünscht schwer wird und somit das Diffie-Hellman-Protokoll nach heutigem Kenntnisstand sehr sicher wird. Aus diesem Grund ist es wichtig, dieses Teilgebiet der Mathematik zu erläutern und schließlich ein Beispiel für eine sichere Methode des Diffie-Hellman-Protokolls vorzustellen.

1.5.1 Mathematische Grundlagen

Wir erinnern noch einmal an die in Kapitel 1.3 vorgestellten Grundlagen.

Definition 1.5.1 Polynome:
Sei K ein Körper, $n \in \mathbb{N}$ und $a \in K$. Ein **Polynom** über K ist ein Ausdruck der Form:

$$f(x) \; = \; a_n \cdot x^n \; + a_{n-1} \cdot x^{n-1} \; + \ldots + \; a_1 \cdot x \; + \; a_0$$

.

Der Grad des Polynoms wird durch n angegeben. Weiter gilt, dass die Menge der Polynome über einem Körper K einen Ring mit Polynomaddition und Polynommultiplikation bilden. Den so genannten Polynomring im folgenden denn bezeichnet mit $K[X]$. Darüberhinaus bezeichnet das Wort $a_n \cdot a_{n-1} \cdot \ldots \cdot a_0$ das charakteristische Wort des Polynoms. Es existiert weiter eine bijektive Abbildung zwischen Polynomen und ihren charakteristischen Wörtern.

Proposition 1.5.1 Charakteristik eines Körpers:
Die Charakteristik eines Körpers ist entweder 0 oder die Anzahl der Elemente des Körpers. Im Fall von $\mathbb{Z}_p$ ist die Charakteristik $p-1$.

Definition 1.5.2 Erweiterung:
Seien K und L Körper. Wenn ein Körper Homomorphismus von K nach L besteht, dann ist L eine **Erweiterung** von K, geschrieben L/K.

Definition 1.5.3 Algebraisch abgeschlossen:
Ein Körper K heißt **algebraisch abgeschlossen**, wenn eine der folgenden äquivalenten Bedingungen gilt:

1. Jedes Polynom $f \in K[X]$ vom Grad $n \geq 1$ hat mindestens eine Nullstelle in K.
2. Jedes $f \in K[X]$ vom Grad $n \geq 1$ zerfällt in vollständige Linearfaktoren.

Definition 1.5.4 Algebraischer Abschluss:
Die Körpererweiterung $\overline{K}$ eines Körpers K heißt **algebraischer Abschluss** von K, wenn eine der folgenden äquivalenten Bedingungen erfüllt ist.

1. $\overline{K}|K$ ist algebraisch und $\overline{K}$ ist algebraisch abgeschlossen.
2. $\overline{K}$ ist algebraisch abgeschlossen und minimale Erweiterung von K.

Galois Feld:
Ein Galois Feld GF ist ein Körper, der aus einer Menge mit endlich vielen Elementen besteht, auf der die Operationen Addition und Multiplikation definiert sind. Mit diesem Hilfsmittel haben wir nun einen Körper mit endlich vielen Elementen. Dies benötigen wir für die sinnvolle Konstruktion unserer elliptischen Kurven.
$GF(2^n)$: Es seien n eine natürliche Zahl und $p(x)$ ein unzerlegbares Polynom mit Grad n und binären Koeffizienten. $GF(2^n)$ ist nun die Menge aller Polynome vom Grad $< n$ mit binären Koeffizienten. Weiter gelten folgende Operationen. Alle Polynome werden bitweise modulo 2 addiert und multipliziert wird gemäß den gegebenen Rechenregeln, mit anschließender Reduktion modulo $p(x)$. Auf diese Weise wird das Galois Feld zu einem endlichen Körper mit 2^n Elementen. Ein weiteres Beispiel ist das Galois-Feld GF(p), wobei p eine Primzahl ist. Dieses Galois-Feld, auch Polynomkörper genannt, wird durch die beiden Gruppen $\mathbb{Z}_p^+$ und $\mathbb{Z}_p^* \backslash \{0\}$ gebildet. Wobei beide Gruppen aus der Menge der Zahlen von 0 bis $p-1$ bestehen. Weiterhin ist $\mathbb{Z}_p^+$ eine Gruppe bezüglich der Modulo-Addition und $\mathbb{Z}_p^* \backslash \{0\}$ ist eine Gruppe bezüglich der Modulo-Multiplikation. Im Folgenden benutzen wir als Beispiel für unser Galois-Feld $GF(p)$ den Körper $\mathbb{Z}_p$.

1.5.2 Einführung in elliptische Kurven

Unser Ziel ist es, mit Hilfe der Punkte einer elliptischen Kurve eine abelsche Gruppe zu definieren, die für das Diffie-Hellman-Protokoll gut geeignet ist. Zunächst definieren wir

eine allgemeine, algebraische Kurve.

Definition 1.5.5 Algebraische Kurve:
Sei f ein Polynom. Eine algebraische Kurve über einen Körper K mit zwei Variablen x und y und Koeffizienten aus K ist definiert duch E:{x,y $\in K|$ $f\ (x,y)\ =\ 0$}. Der Grad des Polynoms f wird als Grad der Kurve bezeichnet.

Kommen wir nun zum Spezialfall einer Kurve, nämlich den elliptischen Kurven. Wir beginnen mit einer allgemeinen Definition.

Definition 1.5.6 Elliptische Kurve:
Eine elliptische Kurve über einem Körper K ist eine Kurve E vom Grad 3 über K, welche durch eine Gleichung der Form

$$E: y^2\ +\ a_1xy\ +\ a_3y\ =\ x^3\ +\ a_2x^2\ +\ a_4x\ + a_6$$

mit $a_1, a_2, a_3, a_4, a_6 \in K$ gegeben ist. Diese Gleichung wird auch als lange Weierstraß-Gleichung bezeichnet.

Ab sofort bezeichnet E unsere elliptischen Kurven. Wir können weiter unsere Definition der elliptischen Kurven mit Hilfe der langen Weierstraß-Gleichung noch vereinfachen. Hierzu benötigen wir den Begriff der Charakteristik eines Körpers K, die mit der Notation $char(K)$ eingeht. Die Charakteristik ist eine Kennzahl des Körpers. Sie gibt an, wie oft man die im Körper enthaltene Zahl 1 aufaddieren muss, um als Ergebnis 0 zu erhalten. Mit Hilfe der Charakteristik kommen wir zu folgendem Lemma.

Lemma 1.5.1:
Sei E über K eine elliptische Kurve. Sei $char(K) \neq 2, 3$. Dann lassen sich elliptische Kurven schreiben als

$$E: y^2\ =\ x^3\ + a \cdot x\ +\ b.$$

Diese Gleichung bezeichnet man als kurze Weierstraß-Gleichung.
Weiter ist diese Gleichung für unsere Zwecke vollkommen ausreichend, da sie alle Eigenschaften erfüllt, die wir benötigen, wie wir noch sehen werden. Eine Einschränkung muss jedoch gelten, damit unsere kurze Weierstraß Gleichung überhaupt eine elliptische Kurve definiert. Es muss gelten, dass $4a^3\ +\ 27b^2\ \neq\ 0$ ist, ansonsten wäre unsere Kurve nicht glatt.[22] Eine Eigenschaft, die wir aber benötigen. Jetzt können wir die Menge der K-rationalen Punkte von E geschrieben $E(K)$ definieren, als die Menge der Lösungen$(\zeta, \eta) \in K \times K$ unserer kurzen Weierstraß-Gleichung. Nun brauchen wir noch das neutrale Element auf unserer Kurve. Das ist für die spätere Komposition unserer Gruppe wichtig. Das neutrale Element bezeichnen wir mit $\mathbb{O}$ und nehmen weiter $\mathbb{O}$ als "im Unendlichen" liegend an. Wir setzen also

$$E(K)\ =\ \{(\zeta, \eta) \in K \times K | \eta^2\ =\ \zeta^3\ +\ a\zeta\ + b\} \cup \{\mathbb{O}\}.$$

[22]Glatt meint das unsere Kurve stetig differenzierbar ist, mit einer nicht verschwindenden Ableitung. Anschaulich heißt das, dass unsere Kurven keine spitzen Ecken oder abrupte Wendungen haben.

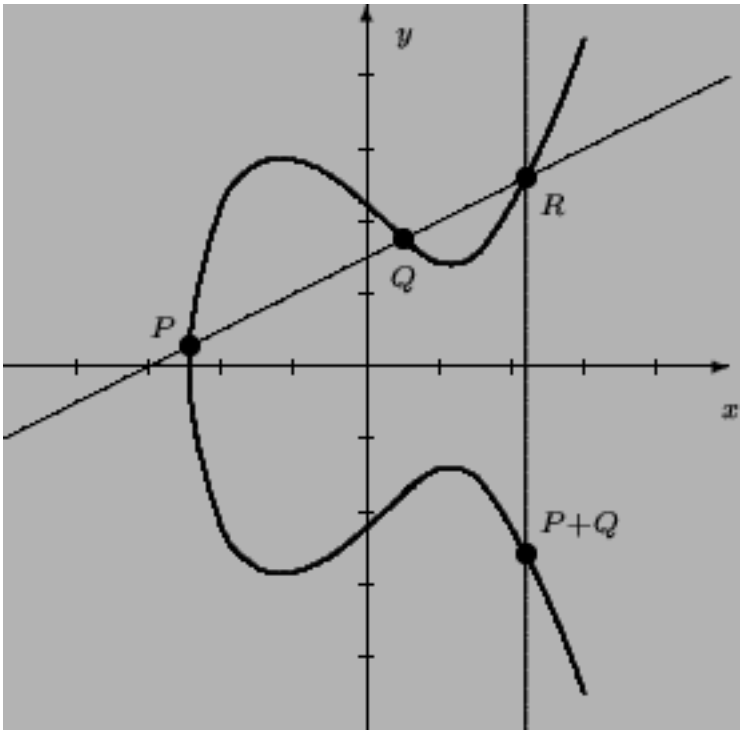


Abbildung 1.4: Kompositionsgesetz für elliptische Kurven

Nun kommen wir zum eigentlichen Phänomen der elliptischen Kurven, das für uns die entscheidende Rolle spielt. Die rationalen Punkte einer elliptischen Kurve bilden auf natürliche Weise eine abelsche Gruppe. Weiter bildet die Summe dreier Schnittpunkte auf einer Graden durch E unser $\mathbb{O}$. Dabei ist auf die richtige Vielfachheit der Schnittpunkte zu achten. Es gilt, dass Tangenten in einem Punkt die Vielfachheit 2 ergeben und Wendetangenten die Vielfachheit 3. Schließlich muss man $\mathbb{O}$ im Falle einer senkrechten Geraden als dritten Schnittpunkt interpretieren. Dies ergibt sich sich ganz natürlich, wenn man E als projektive Kurve betrachtet.[23] Dieses so genannte Gruppengesetz wollen wir nun etwas ausführlicher beschreiben.

Sind P, Q Punkte auf einer elliptischen Kurve E, so existiert eine Gerade durch diese beiden Punkte, welche E in genau einem weiteren Punkt schneidet. Dieser dritte Schnittpunkt sei R genannt. Im Fall von $P = Q$ sei die Gerade die Tangente in P an E. Erstellt man nun eine weitere, diesmal vertikale Gerade durch R, so schneidet diese die Kurve E wiederum in genau einem Punkt $\grave{R}$. Dieser Punkt bildet nun die Addition der beiden Punkte P und Q auf der elliptischen Kurve E[24]. Durch eben diese Addition kann nun die schon angesprochene abelsche Gruppe definiert werden. Diesen Schritt wollen wir nun formal durchführen und beginnen mit Komposition $\oplus$ auf E. Dazu folgt ein Kompositionsgesetz.

[23]Projektiv bedeutet, dass die Kurve durch homogene Koordinaten (x,y,z) beschrieben wird, wobei x, y und z nicht gleichzeitig 0 werden dürfen und zwei Punkte als gleich aufgefasst werden, wenn sie durch Multiplikation einer von 0 verschiedenen Zahl auseinander hervorgehen

[24]Zum besseren Verständnis der nötigen Operation für die Addition auf elliptischen Kurven, siehe auch Abbildung 1.4.

Kompositionsgesetz 1.5.1:
Seien $P, Q \in E$, L die Gerade welche P und Q verbindet und R ein dritter Punkt auf dem Schnittpunkt von L mit E. Im Falle $P = Q$ sei L die Tangente in P an E. Weiter sei $\grave{L}$ die vertikale Gerade zwischen R und $\mathbb{O}$. Dann ist $P \oplus Q$ der Punkt, so dass $\grave{L}$ die Kurve E in R, $\mathbb{O}$ und $P \oplus Q$ schneidet.
L und $\grave{L}$ sind in unserer Abbildung zu sehen, falls P und Q unterschiedlich sind. Jetzt müssen wir aber noch zeigen, dass wir mit Hilfe dieser Komposition auch wirklich eine abelsche Gruppe durch die Menge der Punkte auf E definieren können. Dazu betrachten wir die Eigenschaften unseres Kompositionsgesetzes.

Satz 3.2:
Das Kompositionsgesetz besitzt folgende Eigenschaften:

- Falls eine Gerade die elliptische Kurve E in den Punkten P, Q, R schneidet, gilt
$$(P \oplus Q) \oplus R = \mathbb{O}$$
- $P \oplus \mathbb{O} = P$ für alle $P \in E$
- Sei $P \in E$. Dann existiert ein Punkt $\ominus P$, so dass gilt
$$P \oplus (\ominus) P = \mathbb{O}$$
- $P \oplus Q = Q \oplus P$, für alle $P, Q \in E$
- Seien $P, Q, R \in E$. Dann gilt
$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$$
- Sei E definiert über K. Dann ist
$$E(K) := \{(x, y) \in K^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathbb{O}\}$$

Die Beweise sparen wir uns, dem interessierten Leser sei hierfür [7] aus dem Literaturverzeichnis nahegelegt. Aus den aufgeführten Eigenschaften geht hervor, dass unser Kompositionsgesetz gerade eine abelsche Gruppe (E,$\oplus$) mit $\mathbb{O}$ als neutralem Element definiert.

1.5.3 Elliptische Kurven in der kryptographischen Praxis

In der kryptographischen Praxis sind Kurven die über einer unendlichen Mengen definiert sind wenig sinnvoll. Die Kryptographie wünscht sich eine endliche Anzahl an Punkten um dem diskreten Logarithmusproblem die gewünschte Härte zu geben. Weiterhin führt die Berechnung mit diskreten Mengen zu glatten Ergebnissen und ist verhältnismäßig schnell durchzuführen. Im Gegensatz dazu liefern z.B. elliptische Kurven, bei der Berechnung, über dem Körper der reellen Zahlen Rundungsfehler und einen erheblichen Mehraufwand bezüglich der benötigten Laufzeit. Der Körper $GF(p)$ hat sich in diesem Zusammenhang

als äußerst wirkungsvoll herausgestellt. Er liefert uns alle nötigen Parameter um unsere elliptische Kurve zu beschreiben. Weiterhin ist auch der Körper $GF(2^n)$ sehr gut geeignet. In der Praxis verwendet man $GF(p)$ eher bei Softwareanwendungen und $GF(2^n)$ bei Hardwareanwendungen. Mit Hilfe elliptischer Kurven über diesen Gruppen ist es nun möglich zu einem gegebenen Punkt P auf der Kurve und einer beliebigen natürlichen Zahl m das Produkt $m \cdot P$ zu bilden. Dieses Produkt ist auf folgende Art und Weise

$$m \cdot P \; = \; \underbrace{P \; + \; P \; + \; \ldots \; + \; P}_{m-mal}$$

definiert.

1.5.4 Diffie - Hellman und elliptische Kurven

Wendet man das Diffie-Hellman-Protokoll auf elliptischen Kurven an, ist folgender Weg zu gehen. Vorab muss die elliptische Kurvengleichung zwischen Alice und Bob ausgetauscht werden. Darüberhinaus müssen sich beide auf einen Punkt $P \in E$ einigen. Dabei muss auch die Ordnung r des Punktes bekannt sein. Schließlich sind wir jetzt bei dem entscheidenden Schritt angelangt unser bisher erlangtes Wissen derart zu verknüpfen, dass wir genau erläutern wie und warum unser Diffie-Hellman-Verfahren so gut mit elliptischen Kurven funktioniert. Dazu seien nun die einzelnen Schritte die unser Diffie-Hellman-Verfahren über elliptischen Kurven durchläuft explizit erläutert.

1. Bob und Alice müssen sich auf eine öffentliche Gruppe $\mathbb{Z}_p$ einigen und auf eine ganze Zahl q.

2. Nun müssen r_a, von Alice und r_b, von Bob, zufällig gewählt werden, wobei gelten soll, $r_a, r_b \in \{1, 2, \ldots, q-1\}$.

3. Jetzt muss Alice R_a bestimmen, mit

   $$R_a \; = \; r_a P$$

   und sendet R_a dann an Bob. Bob muss R_b bestimmen, mit

   $$R_b \; = \; r_b P$$

   und sendet dann seinerseits R_b an Alice. Es sei erwähnt, dass r_a *und* r_b geheim bleiben.

4. Nun können Alice und Bob leicht $R = r_a r_b P \; = \; r_b r_c P$ berechnen. Anzumerken ist an dieser Stelle, dass R_a und R_b leicht zu berechnen sind. Es erfordert jeweils r_a bzw. r_b Additionen. Aus ihnen und dem Kurvenpunkt P dann r_a oder r_b zu berechnen gestaltet sich jedoch als schwierig. Die Division zweier Kurvenpunkte ist nämlich nicht existent und darüberhinaus ist bisher noch kein effizienter Algorithmus bekannt, der auf einem anderen Weg r_a bzw. r_b bestimmen kann. Dieses Problem, also in unserem speziellen Fall aus $R_b \; = \; r_b P$ mit Kenntnis von R_b und P r_b zu bestimmen, bezeichnet man auch als das **diskrete Logarithmusproblem über elliptischen Kurven**.

1.6 Rechenbeispiel

Für unser Rechenbeispiel legen wir erst einmal fest, welche Parameter wir brauchen.

- $E(\mathbb{Z}_p)$: $E(\mathbb{Z}_p)$ ist wieder unsere elliptische Kurve über $\mathbb{Z}_p$, berechnet nach der Gleichung
$$E : y^2 = x^3 + ax + b.$$
- p : p gibt die Größe des zugrunde liegenden Körpers $\mathbb{Z}_p$ an.
- a, b: Die Koeffizienten von E.
- P: P soll ein Punkt von E sein.
- r: r enspricht der Ordnung des Punktes P und muss eine Primzahl sein.

Nun wählen wir $p = 149, a = 5, b = 6$, für E folgt daraus, $E : y^2 = x^3 + 5x + 6$. Weiter wählen wir $P = (66, 34)$ mit $r(P) = 71$. Darüberhinaus ist der Wert $\#E(\mathbb{Z}_p) = 142$ für eine abschließende Betrachtung wichtig.[25] Aber nun endgültig zu unserem Beispiel.

1. In unserem Beispiel wählt zunächst Alice ihr persönliches, geheimes $r_a = 18$ und Bob sein persönliches, geheimes $r_p = 69$.

2. Nun berechnet Alice $R_a = r_a P = \underbrace{P + P + \ldots + P}_{r_a - mal} = 18 \cdot (66, 34) = (66, 115)$ und sendet R_a anschließend an Bob. Bob seinerseits berechnet $R_b = r_b P = 69 \cdot (66, 34) = (27, 56)$ und sendet R_b an Alice.

3. Nun berechnen Alice und Bob den geheimen Schlüssel $R = r_a \cdot R_b = r_b \cdot R_a = (27, 93)$

R soll nun als Schlüssel fungieren um Alice und Bob eine sichere Kommunikation zu ermöglichen. In der Praxis gelten Schlüssel jedoch erst als sicher, wenn $\#E(\mathbb{Z}_p) = r \cdot c$ gilt, mit der Bedingung, dass $r > 2^{160}$ und prim ist.

[25] # bezeichnet die Ordnung unserer Kurve $E(\mathbb{Z}_p)$

1.7 Zusammenfassung

Wir haben in dieser Arbeit Grundlagen in einigen Gebieten erhalten, die für das Verständnis der Kryptographie im Allgemeinen und für das Diffie-Hellman-Verfahren mit Hilfe elliptischer Kurven im Besonderen notwendig sind. Im Kapitel 1.2 haben wir die Komplexitätstheorie vorgestellt. Es ist unerlässlich, auf diesem Gebiet Grundlagen zu besitzen, wenn man über die Effizienz von Kryptographieverfahren spricht. Genauer gesagt ist die Komplexitätstheorie ein entscheidendes Hilfsmittel für die Bestimmung der Härte diverser Algorithmen. Für unsere Verfahren in der Kryptographie ist dies deswegen nötig, da es von entscheidender Bedeutung ist, diejenigen Algorithmen zu kategorisieren, die in der Lage sind unser Verfahren zu entschlüsseln. Wir wollen wissen, wie gut die Algorithmen, die unser jeweiliges Problem lösen würden arbeiten um Aussagen darüber treffen zu können, wie sicher unser Verfahren ist.
Im Kapitel 1.3 gehen wir dann genauer auf das zentrale Problem ein, auf dem das Diffie-Hellman-Verfahren beruht. Das diskrete Logarithmusproblem wird ausführlich erläutert, um dann aufzuzeigen, dass es das Diffie-Hellman-Verfahren zu einem sehr sicheren Verfahren macht. Hier ist es dem Leser auch möglich erste tiefere Einblicke in die Mathematik zu erhalten, die auf diesem Themengebiet unerläßlich ist.
Asymmetrische Verschlüsselung ist erst durch Diffie und Hellman möglich gemacht worden. Das Kapitel 1.4 zeigt auf wieso diese neue Art der Verschlüsselung ein solcher Meilenstein für die Kryptographie war. Weiterhin wird in diesem Kapitel das Diffie-Hellman-Verfahren explizit vorgestellt. Man erlangt hier die nötigen Kenntnisse um allgemein zu verstehen, wie das Verfahren funktioniert. Die Theorie des Verfahrens, ohne zu sehr auf die nötige Mathematik einzugehen, ist auch dem weniger erfahrenen Leser gut zugänglich. Im Gegensatz dazu ist das Kapitel 1.5 zusammen mit den mathematischen Grundlagen aus 1.3 ein Sprung ins tiefe Becken der Algebra. Es wurde versucht, sich bei den mathematischen Anteilen auf das Nötigste zu beschränken und vielerorts mit Beispielen zu unterstützen. Trotzdem verlangt gerade die Theorie der elliptischen Kurven einige Grundkenntnisse im Gebiet der Algebra, auf die wir nicht verzichtet haben. Die elliptischen Kurven wurden so wie wir sie benötigen eingeführt um dann aufzuzeigen, wie man mit ihrer Hilfe das Diffie-Hellman-Verfahren anwendet. Hierbei wurde deutlich gemacht, dass es von entscheidender Bedeutung ist, die elliptischen Kurven über Galois-Feldern zu betrachten.
Im Zuge der Arbeit wird deutlich, dass Verfahren, die auf den elliptischen Kurven basieren, eine große Zukunft im Bereich der Smartcards haben. Gerade aufgrund ihres geringen Speicherbedarfs sind sie hierfür wie geschaffen. Es bleibt abzuwarten, in wie weit die Forschung voranschreitet, wenn es darum geht Algorithmen zu entwerfen die das diskrete Logarithmusproblem über elliptischen Kurven eventuell doch lösen. Zur Zeit droht den elliptischen Kurven aus dieser Richtung noch keine Gefahr. Auch die neue Technik der Quantencomputer steckt noch so sehr in den Kinderschuhen, dass Prognosen, bezüglich der zukünftigen Anwendungen von ECC nicht beunruhigend sind.

Literaturverzeichnis

[1] Werner, Annette. *Elliptische Kurven in der Kryptographie*, Springer, 2002.

[2] Eckert, Claudia. *IT-Sicherheit, Konzepte-Verfahren-Protokolle*, Oldenbourg Wissenschaftsverlag, 2007.

[3] Buchmann, Johannes. *Einführung in die Kryptographie*, Springer, 2008.

[4] Schmeh, Klaus. *Kryptografie: Verfahren, Protokolle, Infrastrukturen*, Dpunkt.Verlag, 2007.

[5] Koblitz, Neal. *A Course in Number Theory and Cryptography*, Springer, 1994.

[6] Beutelsbacher, Albrecht; Neumann, Heike B.; Schwarzpaul, Thomas. *Kryptographie in Theorie und Praxis*, Vieweg und Teubner, 2005.

[7] Silverman, Joseph H.. *The Arithmetic of Elliptic Curves*, Springer, 1992.

[8] Washington, Lawrence C.. *Elliptic curves: Number Theory and Cryptography*, CRC Press, 2003.

[9] Frey, Gerhard; Rück, Hans-Georg. *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Mathematics of Computation, Vol. 62, No. 206, 1994.

[10] Lenstra, H.W.. *Factoring integers with elliptic curves*, Mathematical Sciences Research Institute, 1986.

[11] Silverman, J.H.; Tate, J.. *Rational points on elliptic curves*, Springer, 1992.

[12] Rothe, Jörg. *Komplexitätstheorie und Kryptologie: Eine Einführung in Kryptokomplexität*, Springer, 2008.

[13] Cohen, Henry; Frey, Gerhard. *Elliptic and Hyperelliptic Curve Cryptography*, Chapman und Hall und CRC, 2006.

[14] Berghammer, Benedikt. *Berechnung der Weil-Paarung und des tripartiten Diffie-Hellman Schlüssels auf elliptischen Kurven*, Diplomarbeit, München, 2008.

[15] Müller, Volker; Paulus, Sascha *Erzeugung kryptographisch starker elliptischer Kurven*, veröffentlicht in Datenschutz und Datensicherheit, Jahrgang 22, 1998.

[16] *http://www.bsi.de/literat/tr/tr03111/BSI-TR-03111.pdf*

[17] *http://www.elliptische-kurven.de/*

[18] *http://www.ecc-brainpool.org/*

[19] *http://www.cryptool.de/*

[20] *http://www.bsi.de/literat/tr/tr03111/BSI-TR-03111.pdf*

[21] *http://www.uni-koeln.de/rrzk/sicherheit/pgp/win/verschluesselung.html*

Kapitel 2

Trusted Computing

Oliver LAM SHIN CHEUNG

Inhaltsverzeichnis

2.1 Introduction

Businesses, governments, academic institutions, and individual users are becoming increasingly interconnected through a variety of wired and wireless communication networks and with a variety of computing devices. Concerns about the security of communications, transactions, and wireless networks are inhibiting realization of benefits associated with pervasive connectivity and electronic commerce. These concerns include exposure of data on systems, system compromise due to software attack, and lack of user identity assurance for authorization. The latter concern is exacerbated by the increasing prevalence of identity theft.
In addition, as users become more mobile, physical theft is becoming a growing concern. Users and IT organizations need the industry to address these issues with standards- based security solutions that reduce the risks associated with participation in an interconnected world while also ensuring interoperability and protecting privacy. Standardization will also enable a more consistent user experience across different device types.
The Trusted Computing Group (TCG) formed in 2003 to respond to this challenge. The purpose of TCG is to develop, define, and promote open, vendor-neutral industry specifications for trusted computing. These include hardware building block and software interface specifications across multiple platforms and operating environments. Implementation of these specifications will help manage data and digital identities more securely, protecting them from external software attack and physical theft. TCG specifications can also provide capabilities that can be used for more secure remote access by the user and enable the user's system to be used as a security token.
This document will provide an overview of Trusted Computing and in peculiar an insight view of the Trusted Platform Module (TPM). First we will start with a presentation of the Trusted Computing Group (TCG) that is the organisation developing the specifications of the TPM, and then we will describe the different elements that characterize the TPM and finally discuss the different applications of this new technology.

2.2 Trusted Computing Group

2.2.1 Presentation

The Trusted Computing Group is a not-for-profit corporation with international membership and broad industry participation. The purpose of TCG is to develop, define, and promote open specifications for trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices. By using the building blocks and software interfaces defined by TCG specifications, the industry is addressing a range of security needs without compromising functional integrity, privacy, or individual rights.
TCG was created with an organization structure and governance model, as defined by the TCG bylaws, which is similar to many other computing industry standard bodies. This includes the following:

- An open membership model with multiple membership level
- A board of directors consisting of Promoters and elected Contributor members
- Multiple work groups that are open to Promoter and Contributor members and seek active participation by these members
- A reciprocal reasonable and non-discriminatory (RAND) patent licensing policy between the members

This structure is designed to enable the expedient development of open, industrystandard specifications with broad industry participation and to foster widespread adoption of the organisation's specifications.
The key deliverables of TCG are hardware and software interfaces specifications, white papers and other materials that facilitate understanding and adoption of the specifications, and marketing programs that promote awareness, capabilities and customer adoption.

2.2.2 Projects

TCG's goal like explained previously is to develop, define, and promote open specifications for trusted computing and security technologies. Thus, they have worked on a secure cryptoprocessor able to store cryptographic keys protecting information on a PC. Here are the TCG policies that have impacted specifications development for the TPM:

- Open platform development model - TCG is committed to preserving the open development model that enables any party to develop hardware, software, or systems based on TCG specifications. Further, TCG is committed to preserving the freedom of choice that consumers enjoy with respect to hardware, software, and platforms.
- Platform owner and user control - TCG is committed to ensuring owners and users of computing platforms remain in full control of their computing platform and to requiring platform owners to opt-in to enable TCG features.
- Privacy effect of TCG specifications - TCG is committed to ensure that TCG specifications provide for an increased capability to secure personally identifiable data. TCG currently offers specifications for the hardware Trusted Platform Module, which is in widespread deployment - TPMs are available from a number of vendors and implemented in tens of millions of PCs and other systems.

To extend the specifications beyond the PC, TCG has also work groups for servers, mobile devices, storage, infrastructure and peripherals. A specification for network access control, Trusted Network Connect, has been available to industry for several years, and products to implement it are available from a number of networking equipment and applications providers.
TCG also has released a specification for trusted servers. As with other specifications, this one supports a number of server types and form factors, and as implemented in servers

shipping from several vendors, can help protect financial and other sensitive transactions and support Trusted Computing clients already in deployment.
TCG also has started to extend the concepts of the trusted platform to mobile phones. The work group tackling the issue security on handheld systems has published a specification for the Mobile Trusted Module and anticipates additional specifications to follow. The group also will publish a specification to enable storage security.

2.2.3 Summery

TCG and its member companies are answering the need for increased security and trust in computing platforms. The open hardware building block and software interface specifications developed and promoted by TCG will attain this goal through hardware-based cryptographic functions, protected storage of user data and secrets, mechanisms for secure storage and attestation identities.
As security threats increase, trusted computing and security technologies will evolve. TCG will continue to look for opportunities to work with the industry to make meaningful contributions to enhancing the security of the computing environment.

2.3 Trusted Platform Module

2.3.1 Role

The Trusted Platform Module (TPM) is a hardware component that securely stores digital keys, certificates and passwords. TPMs protect encryption keys and digital signature keys to maintain data confidentiality. TPM chips are designed to protect key operations and other security tasks that would otherwise be performed on unprotected interfaces in unprotected communications. Especially important, TPMs are specifically designed to protect platform and user authentification information and uncrypted keys from software-based attacks or physical attacks. In order to provide these trusted features the TPM is based on two key principles:

- A protected capability is one whose correct operation is necessary in order for the operation of the TCG Subsystem to be trusted.
- A shielded location is an area where data is protected against interference and prying, independent of its form.

TPM protected capabilities are the only means from outside the TPM to access the information represented by data in TPM shielded locations. Moreover a TPM shouldn't facilitate the alteration of TPM protected capabilities, except by TPM protected capabilities. Only plain text data of the system that can be used without violating security or privacy are accessible to exterior actions. This exception is valuable because it approves use of TPM resources by vendor-specific commands in particular circumstances.

Summery

Critical information is stored in shielded locations which are only accessible by TPM protected capabilities which themselves can only be altered by other protected capabilities. Only data that can be used without violating security or privacy are available to external applications. This is the philosophy of the TPM.

2.3.2 Principle

To better understand the main feature of the TPM, that is the attestation principle, we will first present the characteristic of the TPM in the version 1.1b, and then explain the evolutions done in the version 1.2. The technical information concerning the keys will be presented in part [2.3.3].

Direct Anonymous Attestation

Attestation is an important TPM function with significant privacy implications. The primary goal of the attestation process is to reliably communicate, at the TPM Owner's request, information about the static capabilities of a computing platform that contains a TPM to a remote party.
The TPM v1.1b attestation process was defined to ensure that the remote party that receives the attestation could have a reasonable expectation that the information presented is valid. Cryptographic techniques based on a platform-unique cryptographic key are used to achieve this design goal. The platform-unique key is called the Endorsement Key or EK.
During the initial development of the TPM specification the contributors to the specification were aware that the EK presented privacy difficulties. While the private part of the EK (EKpr) never leaves the TPM, disclosure of the public part of the EK (EKpu) is necessary.
Free disclosure of the EKpu was understood to be unacceptable, and the Trusted Third Party (TTP) model was developed. Under this model the platform uses the EKpu only when interacting with a Trusted Third Party as part of a controlled process that results in the generation of a secondary key-pair. This secondary key-pair would be used in the actual attestation process with untrusted parties. This secondary key-pair is called the Attestation Identity Key (AIK).
If a platform were to use only a single AIK, the privacy problems associated with the disclosure of the EK to un-trusted parties would just be substituted for the similar problems with the AIK. To address this concern the TTP model provides the ability for users to obtain many different AIKs. To ensure anonymity, different AIKs should be used with different parties. Where appropriate a user could even use a different AIK each time attestation is made.
So long as the relationship between the multiple AIKs and the individual primary EK is kept confidential, the use of multiple AIKs solves the privacy problems potentially associated with disclosure of the EK_{pu}.

Evolutions

While the Trusted Third Party model for AIK creation can achieve the necessary privacy characteristics, it still requires the disclosure of the EK_{pu} to the TTP. The new Direct Anonymous Attestation (DAA) method for AIK creation was developed as a means to achieve enhanced privacy by eliminating the need to share the EK_{pu} with the TTP. The DAA was also designed to allow a computing platform to directly attest to platform characteristics without the use of the TTP as an intermediary.
DAA is based on a family of advanced cryptographic techniques known as Zero Knowledge Proofs (ZKP). The details and principles of the ZKP techniques used in DAA are beyond the scope of this paper. From a privacy perspective, the critical new feature provided by DAA is its ability to convince a remote entity (the "Verifier") that a particular TPM is a valid TPM without disclosing the EKpu or any other unique identifier.
To use the DAA protocol, the TPM must first generate a set of DAA- credentials. To do this, the TPM (under TPM Owner control) interacts with an "Issuer". This can be done multiple times. The DAA-credentials are generated in an interaction between the TPM and the Issuer using a TPM-unique secret that remains within the TPM. This TPMunique secret is used in every DAA-credential creation, and is: distinct from, but analogous to, the EK.
The DAA credentials are used during an interaction defined by the DAA protocol with a second party called the Verifier. Under the right circumstances, at the end of the protocol the Verifier can determine if the TPM contains a valid set of DAA- credentials from a particular Issuer, but does not have specific knowledge of identifying the TPM from among others that also have valid DAA-credentials.

2.3.3 Architecture

Components

The TPM contains several different key generators and security devices. The following scheme is a block diagram that shows the major components of a TPM:

Input and Output

The I/O component (C0) manages information flow over the communications bus. It performs protocol encoding/decoding suitable for communication over external and internal buses and routes messages to appropriate components. The I/O component enforces access policies associated with the Opt-In component as well as other TPM functions requiring access control.

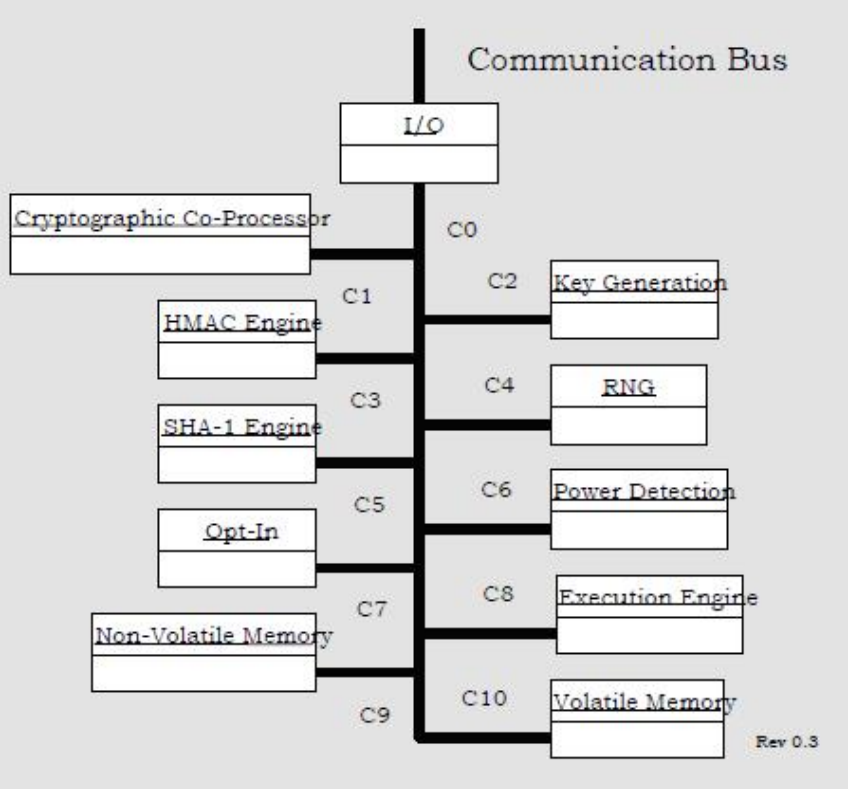


Abbildung 2.1: TPM Component Architecture

Cryptographic Co-Processor

The cryptographic co-processor (C1) implements cryptographic operations within the TPM. The TPM employs conventional cryptographic operations in conventional ways. Those operations include the following:

- Asymmetric key generation (RSA)
- Asymmetric encryption/decryption (RSA)
- Hashing (SHA-1)
- Random Number generation (RNG)

The TPM uses these capabilities to perform generation of random data, generation of asymmetric keys, signing and confidentiality of stored data. Note: All storage keys are of strength equivalent to a 2048 bits RSA key or greater.

Key generation

The key generation component (C2) creates RSA key pairs and symmetric keys. The generating function is a protected capability and the private keys are held in shielded locations.

HMAC Engine

The HMAC Engine (C3) provides two pieces of information to the TPM: proof of knowledge of the AuthData (Authentification and Authorization Data, cf.[B.2.3]) and proof that the request arriving is authorized and has no modifications made to the command in transit.

Random Number Generator

The Random Number Generator (RNG) component (C4) is the source of randomness in the TPM. The TPM uses these random values for nonces (= number used once), key generation, and randomness in signatures. The RNG capability is a TPM-protected capability with no access control.
The RNG consists of a state-machine that accepts and mixes unpredictable data and a post-processor that has a one-way function (e.g. SHA-1). The idea behind the design is that a TPM can be good source of randomness without having to require a genuine source of hardware entropy.
Example:
The state-machine can have a non-volatile state initialized with unpredictable random data during TPM manufacturing before delivery of the TPM to the customers. The state-machine can accept, at any time, further (unpredictable) data, or entropy, to salt the random number. Such data comes from hardware or software sources - for example; from thermal noise, or by monitoring random keyboard strokes or mouse movements. The RNG requires a reseeding after each reset of the TPM. A true hardware source of entropy is likely to supply entropy at a higher baud rate than a software source.
The RNG for the TPM will consist of the following components:

1. Entropy source and collector (thermal noise, keyboard strokes, ...)

2. State register: keeps startup state and current state in two registers

3. Mixing function: creates the output with information from the Entropy source and collector and the state register. Each use of the mixing function must affect the state register.

SHA-1 Engine

The SHA-1 (C5) hash capability is primarily used by the TPM, as it is a trusted implementation of a hash algorithm. The hash interfaces are exposed outside the TPM to support Measurement taking during platform boot phases and to allow environments that have limited capabilities access to a hash functions.

Power Detection

The power detection component (C6) manages the TPM power states in conjunction with platform power states. In addition it also supports physical presence assertions. The TPM may restrict command-execution during periods when the operation of the platform is physically constrained.
Example:
In a PC, operational constraints occur during the power-on self-test (POST) and require Operator input via the keyboard. The TPM might allow access to certain commands while in a constrained execution mode or boot state. At some critical point in the POST process, the TPM may be notified of state changes that affect TPM command processing modes.

Opt-In

The Opt-In component (C7) provides mechanisms and protections to allow the TPM to be turned on/off, enabled/disabled, activated/deactivated. The Opt-In component maintains the state of persistent and volatile flags and enforces the semantics associated with these flags.
The setting of flags requires either authorization by the TPM Owner or the assertion of physical presence at the platform. The platform's manufacturer determines the techniques used to represent physical-presence. The guiding principle is that no remote entity should be able to change TPM status without either knowledge of the TPM Owner or the Operator is physically present at the platform. Physical presence may be asserted during a period when platform operation is constrained such as power-up.

Execution Engine

The execution engine (C8) runs program code to execute the TPM commands received from the I/O port. The execution engine is a vital component in ensuring that operations are properly segregated and shield locations are protected.

Non-Volatile Memory

Non-volatile memory component (C9) is used to store persistent identity and state associated with the TPM. The NV area has set items (like the EK) and also is available for allocation and use by entities authorized by the TPM Owner.

Registers

There are two kinds of registers in the TMP:

- Data Integrity Register (DIR): 160 bits register part of the NV storage area.
- Platform Configuration Register (PCR): 160 bit storage location for discrete integrity measurements.

2.3.4 Keys and Signatures

Endorsement Key Creation

The Endorsement Key (EK) is a 2048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip, as occurs during the TPM_TakeOwnership command.
This key is used to allow the executions of secure transactions: every TPM is required to sign a random number, using a particular protocol created by the trusted computing group (the direct anonymous attestation protocol) in order to ensure its compliance of the TCG standard and to prove its identity; this makes it impossible for a software TPM emulator, with a self-generated EK, to start a secure transaction with a trusted entity. The TPM should be designed to make the extraction of this key by hardware analysis hard, but tamper-resistance is not a strong requirement.

Attestation Identity Keys

An Attestation Identity Key (AIK) is an alias for the Endorsement Key (EK). The EK cannot perform signatures for security reasons and due to privacy concerns, that's why AIK have been created. An AIK is a signature key, and is never used for encryption. It only signs information generated internally by the TPM.
The AIK is an asymmetric key pair. For interoperability, the AIK is an RSA 2048-bit key. The TPM must protect the private portion of the asymmetric key and ensure that the value is never exposed, as for the EK. Generation of an AIK can occur anytime after establishment of the TPM Owner. Thus, a virtually unlimited number of AIK can be created.

Authentification and Authorization Data

Each TPM object that does not allow "public" access contains a 160-bit shared secret. This shared secret is enveloped within the object itself. The TPM grants use of TPM objects based on the presentation of the matching 160-bits using protocols designed to provide protection of the shared secret. This shared secret is the AuthData.
Neither the TPM, nor its objects (such as keys), contain access controls for its objects. If a subject presents the AuthData, that subject is granted full use of the object based on the object's capabilities, not a set of rights or permissions of the subject.
From the perspective of the TPM looking out, this AuthData is its sole mechanism for authenticating the owner of its objects, thus from its perspective it is authentification

data. However, from the application's perspective this data is typically the result of other functions that might perform authentications or authorizations of subjects using higher level mechanisms such as OS login, file system access, etc. Here AuthData is a result of these functions so in this usage, it authorizes access to the TPM's objects.
A wide-range of objects uses AuthData. It is used to establish platform ownership, key use restrictions, object migration and to apply access control to opaque objects protected by the TPM.

Transport Sessions and Authorization Protocols

The purpose of the authorization protocols and mechanisms is to prove to the TPM that the requestor has permission to perform a function and use some object. The proof comes from the knowledge of a shared secret.
AuthData is available for the TPM Owner and each entity (keys for example) that the TPM controls. The AuthData for the TPM Owner and the SRK are held within the TPM itself and the AuthData for other entities are held with the entity. The TPM treats knowledge of the AuthData as complete proof of ownership of the entity.
There are three protocols to securely pass a proof of knowledge of AuthData from requestor to TPM; the "Object-Independent Authorization Protocol" (OIAP), the "Object-Specific Authorization Protocol" (OSAP) and the "Delegate-Specific Authorization Protocol" (DSAP). The OIAP supports multiple authorization sessions for arbitrary entities. The OSAP supports an authentication session for a single entity and enables the confidential transmission of new authorization information. The DSAP supports the delegation of owner or entity authorization.
New authorization information is inserted by the "AuthData Insertion Protocol" (ADIP) during the creation of an entity. The "AuthData Change Protocol" (ADCP) and the "Asymmetric Authorization Change Protocol" (AACP) allow the changing of the AuthData for an entity. The protocol definitions allow expansion of protocol types to additional TCG required protocols and vendor specific protocols.

2.3.5 Operations

Through the course of TPM operation, it may enter several operational modes that include power-up, self-test, administrative modes and full operation. This section describes TPM operational states and state transition criteria. Where applicable, the TPM commands used to facilitate state transition or function are included in diagrams and descriptions.

TPM Initialization & Operation State Flow

Here is the basic TPM Initialization Operation process: TPM_Init transitions the TPM from a power-off state to one where the TPM begins an initialization process. TPM_Init could be the result of power being applied to the platform or a hard reset.
TPM_Init sets an internal flag to indicate that the TPM is undergoing initialization. The

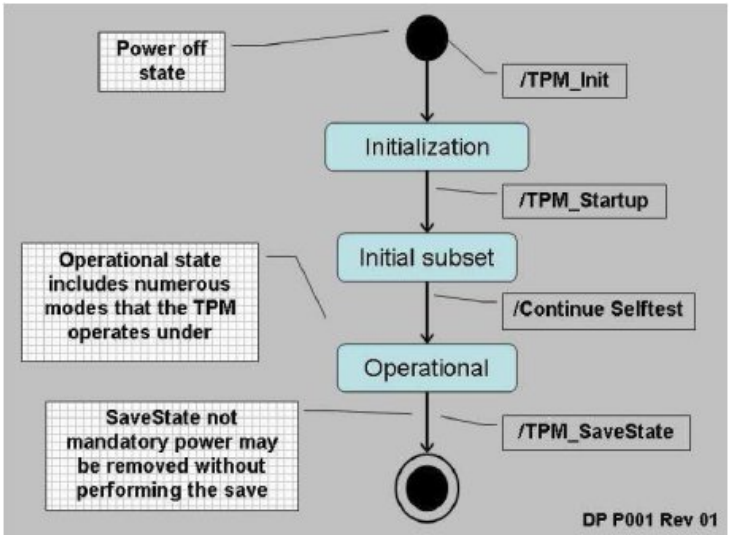


Abbildung 2.2: TPM Operational States

TPM must complete initialization before it is operational. The completion of initialization requires the receipt of the TPM Startup command.
The TPM is not fully operational until all of the self-tests are complete. Successful completion of the self-tests allows the TPM to enter fully operational mode.
The TPM transitions out of the operational mode by having power removed from the system. Prior to the exiting operational mode, the TPM prepares for the transition by executing the TPM_SaveState command. There is no requirement that TPM_SaveState execute before the transition to power-off mode occurs.

Self-Test Modes

After initialization the TPM performs a limited self-test. This test provides the assurance that a selected subset of TPM commands will perform properly. The limited nature of the self-test allows the TPM to be functional in as short of time as possible. The commands enabled by this self-test are:

TPM_SHA1xxx Enabling the SHA-1 commands allows the TPM to assist the platform startup code. The startup code may execute in an extremely constrained memory environment and having the TPM resources available to perform hash functions can allow the measurement of code at an early time. While the hash is available, there are no speed requirements on the I/O bus to the TPM or on the TPM itself so use of this functionality may not meet platform startup requirements.

TPM_Extend Enabling the extend, and by reference the PCR, allows the startup code to perform measurements. Extending could use the SHA-1 TPM commands or perform the hash using the main processor.

TPM_Startup This command must be available as it is the transition command from the initial environment to the limited operational state.

TPM_ContinueSelfTest This command causes the TPM to complete the self-tests on all other TPM functions. If TPM receives a command, and the self-test for that command has not been completed, the TPM may implicitly perform the actions of the TPM_ContinueSelfTest command.

TPM_SelfTestFull A TPM may allow this command after initialization, but typically TPM_ContinueSelfTest would be used to avoid repeating the limited self tests.

TPM_GetCapability A subset of capabilities can be read in the limited operation state.

The complete self-test ensures that all TPM functionality is available and functioning properly.

Startup

Startup transitions the TPM from the initialization state to an operational state. The transition includes information from the platform to inform the TPM of the platform operating state. TPM_Startup has three options: Clear, State and Deactivated.
The Clear option informs the TPM that the platform is starting in a "cleared" state or most likely a complete reboot. The TPM is to set itself to the default values and operational state specified by the TPM Owner.
The State option informs the TPM that the platform is requesting the TPM to recover a saved state and continue operation from the saved state. The platform previously made the TPM_SaveState request to the TPM such that the TPM prepares values to be recovered later.
The Deactivated state informs the TPM that it should not allow further operations and should fail all subsequent command requests. The Deactivated state can only be reset by performing another TPM_Init.

Operational Mode

After the TPM completes both TPM_Startup and self-tests, the TPM is ready for operation. There are three discrete states, enabled or disabled, active or inactive and owned or unowned. These three states when combined form eight operational modes.

S1 is the fully operational state where all TPM functions are available. S8 represents a mode where all TPM features (except those to change the state) are off. Given the eight modes of operation, the TPM can be flexible in accommodating a wide range of usage scenarios.

Enable/Disable

A disabled TPM is not able to execute commands that use the resources of a TPM. While some commands are available (SHA-1) the TPM is not able to load keys and perform

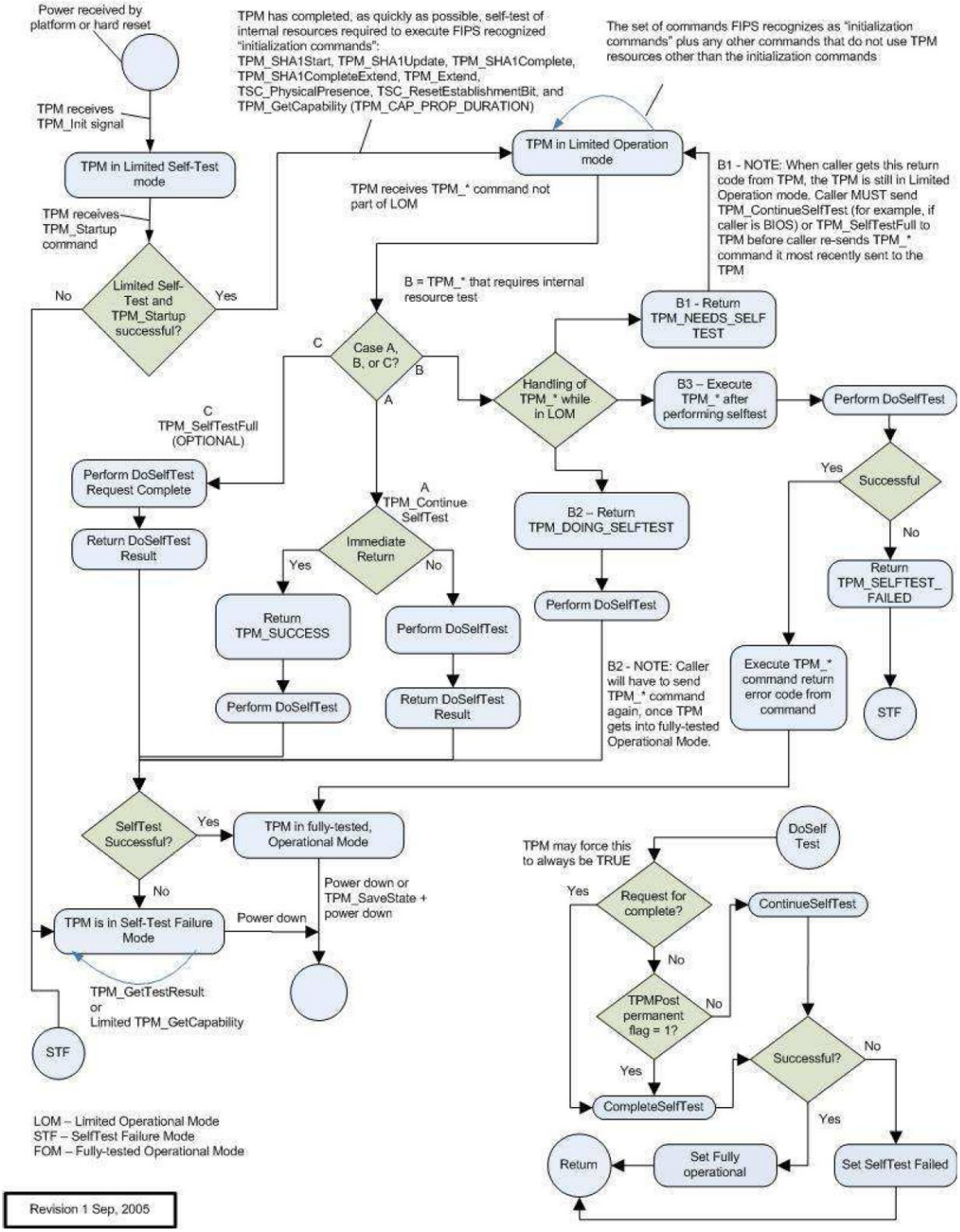


Abbildung 2.3: Self-Test States

Abbildung 2.4: Eigth modes of operation

TPM_Seal and other such operations. These restrictions are the same as for an inactive TPM. The difference between inactive and disabled is that a disabled TPM is unable to execute the TPM_TakeOwnership command. A disabled TPM that has a TPM Owner is not able to execute normal TPM commands.

Activated/Deactivated

A deactivated TPM is not able to execute commands that use TPM resources. A major difference between deactivated and disabled is that a deactivated TPM can execute the TPM_Ownership command.
Deactivated may be used to prevent the attack where a TPM is readied for TPM_TakeOwnership but a remote rogue manages to take ownership of a platform just before the genuine owner, and immediately has use of the TPM's facilities. To defeat this attack, a genuine owner should set disable = = FALSE, ownership = = TRUE, deactivate = = TRUE, execute TPM_takeOwnership, and then set deactivate = = FALSE after verifying that the genuine owner is the actual TPM owner.

Ownership

The owner of the TPM has ultimate control of the TPM. The owner of the TPM can enable or disable the TPM, create AIK and set policies for the TPM. The process of taking ownership must be a tightly controlled process with numerous checks and balances. The protections around the taking of ownership include the enablement status, specific persistent flags and the assertion of physical presence.

2.4 Applications

2.4.1 Applications of Trusted Computing

Next-Generation Secure Computing Base (NGSCB)

Here is the new generation of Microsoft OS based on the TPM's specifications and introduced today by Windows Vista. Formerly called Palladium, this new generation of software architecture is expected to create a more trustworthy computing between distant users.
To put it short, the use of the cryptographic technology of the TPM will be extended to the Operating System to secure the transactions between the processes, between the processes and the RAM, between the processes and the Hard Disk, and between the processes and the input/output peripherals.
For the end-user, this would result into:

- Not allowing documents created by a certain application to be opened by other applications, unless they have the authorization to do so. This feature is called Sealed Storage.
- Blocking unauthorized code to execute, for example viruses, but also applications the user doesn't want.

This new generation of software architecture will introduce more trust between different components (software or hardware) even if they are from distant systems. However Microsoft's new OS has received a lot of critics concerning its application of Digital Rights Management (DRM) on several data and multiple security issues, for example the high amount of warning windows for execution of bad processes that didn't block the user from actually performing them.

BitLocker Drive Encryption

BitLocker Drive Encryption is a full disk encryption feature included with Microsoft's Windows Vista Ultimate, Windows Vista Enterprise and Windows Server 2008 operating systems designed to protect data by providing encryption for entire volumes. When enabled TPM/Bitlocker can also ensure the integrity of the trusted boot path (e.g. BIOS, boot sector, etc.), in order to prevent most offline physical attacks, boot sector malware, etc.
There are three implementation models for BitLocker encryption. Two models require a TPM and a compatible BIOS. A third model does not have the TPM chip requirement:

- Transparent operation mode: This mode exploits the capabilities of the TPM hardware to provide for a transparent user experience - the user logs onto Windows Vista as normal. The key used for the disk encryption is sealed by the TPM chip and will only be released to the OS loader code if the early boot files appear to be

unmodified. The pre-OS components of BitLocker achieve this by implementing a Static Root of Trust Measurement. This mode is vulnerable to a cold boot attack, as it allows a machine to be booted by an attacker.

- User authentification mode: This mode requires that the user provide some authentification to the pre-boot environment in order to be able to boot the OS. Two authentification modes are supported: a pre-boot PIN entered by the user, or a USB key.
- USB key mode (without TPM): The user must insert a USB device that contains a startup key into the computer to be able to boot the protected OS. Note that this mode requires that the BIOS on the protected machine supports the reading of USB device in the pre-OS environment.

These different alternatives have all their pros and cons, for example theft of the USB key for the third mode or cold boot attack for the first one, but Microsoft's BitLocker Drive Encryption is at the moment the only widely spread Trusted Computing application.

Trusted Computing in Free/Open Technologies

Mainstream Linux kernels do not natively recognize TPM chips, and solutions to use them are nearly nonexistent at the moment. However they do exist and it is likely that with the widespread propagation of viruses and other malware, and the ever-increasing security needs of the industry, trusted computing in open source technologies would be very positive. So here is a quick presentation of these technologies.

Trusted Gentoo Trusted Gentoo is a concept proposed by the free/open community to use the TPM. The purpose of this distribution is to provide the user a hardened platform and a choice of utilizing hardware. As we have seen previously, the way that TCG provides a hardened platform is by keeping cryptographic processing in hardware and cryptographic keys out of RAM and disks. It also allows keys to be provided to only user-specified programs.
Trusted Gentoo will give users the choice to set up their hardware to verify that their kernels are loaded on bootup and not some hacked ones that someone installed when they left their computers alone for the night. Trusted Gentoo will give users the ability to store their cryptographic keys (e.g. ssh keys, gnupg keys) in hardware to be released only to those applications the user wants the keys given to. The concept of installing remote trojans to steal crytographic keys and passwords is hard to imagine but not impossible in all setups. The use of TCG architecture will empower the user to deny providing keys to trojans.
Trusted Gentoo gives the choice for users to use the capabilities of the TPM to better protect their systems.

Bear/Enforcer (Darthmouth College) The Enforcer is a Linux Security Module designed to improve integrity of a computer running Linux by ensuring no tampering of the file system. It can interact with TCG hardware to provide higher levels of

assurance for software and sensitive data.
It can check, as every file is opened, if the file has been changed, and take an admin specified action when it detects tampering. The actions can be any combination of log the error, deny access to the file, panic the system, or several operations that work with the TPM.
The Enforcer can also work with the TPM to store the secret to an encrypted loopback file system, and unmount this file system when a tampered file is detected. The secret will not be accessible to mount the loopback file system until the machine has been rebooted with untampered files. This allows sensitive data to be protected from an attacker.
The Enforcer can also bind specific files so that only specific applications can access them (for example, only apache is allowed to access apache's secret ssl key). This means that even if someone compromises your system, the attacker will not be able to steal critical files.
Finally, the Enforcer can make sure that no files added to directories after its database is built are allowed to be accessed.

PERSEUS / TrustedGRUB (Ruhr Universität-Bochum) PERSEUS is a trustworthy computing framework that aims at establishing an open security architecture by efficiently combining existing applications, modern operating system solutions and security technology. Based on the well-approved idea of a security kernel, it solves the problem of the ever increasing complexity and insecurity of commonly used operating systems to build a trustworthy computing base that is secure enough to realise new and innovative business models, particulary in the area of privacy protection and Digital Rights Management (DRM). The modular design of the PERSEUS framework allows the realization of a wide variety of security and functional requirements for desktop systems and servers as well as mobile devices.
Supported by the University of Bochum, PERSEUS couldn't be used with the Enforcer. So another project was carried out in order to extend the trust from the boot to the normal execution of the operating system: TrustedGRUB.
TrustedGRUB is an extension of the original GNU GRUB bootloader. The main feature of TrustedGRUB is the possibility to measure arbitrary files during the boot process and extend the integrity test results into PCR inside TPMs memory.

2.4.2 Possible Applications of Trusted Computing

Digital Rigths Management

Trusted Computing would allow companies to create a Digital rights management system which would be very hard to circumvent. Microsoft's Sealed Storage in the NGSCB for example could easily block the use of a media file (music, video, ...) for any other mediaplayer than Microsoft's or only those authorized by the company, creator of the media file.

Identity Theft Protection

Trusted Computing could be used to help prevent identity theft. Taking online banking as an example, remote attestation could be used when the user is connecting to the bank's server and would only serve the page if the server could produce the correct certificates. Then the user can send his encrypted account number and PIN, with some assurance that the information is private to him and the bank.

Preventing Cheating in Online Games

Trusted computing could be used to combat cheating in online games. Some players modify their game copy in order to gain unfair advantages in the game; remote attestation, secure I/O and memory curtaining could be used to verify that all players connected to a server were running an unmodified copy of the software.
This is especially true with game modifications designed to enhance player ability or automate certain task. For example, a user might want to install an auto aiming bot in shooter games, or a harvesting bot in a strategy game. Since there is no way for the game server to remotely determine if the commands are given by a human being or a program, the proposed solution is to certify the code the player's computer is running.

Protection from viruses and spyware

Digital signature of software will allow users to identify applications modified by third parties that could add spyware to the software. For example, a website offers a modified version of a popular instant messenger that contains spyware as a drive-by download. The operating system could notice the lack of a valid signature for these versions and inform the user that the program has been modified, although this leaves open the question of who determines if a signature is valid.
Trusted computing might allow increased protection from viruses. However, Microsoft has denied that this functionality will be present in its NGSCB architecture. A possible improvement in virus protection would be to allow antivirus vendors to write software that could not be corrupted by virus attacks. However, as with most advanced uses of Trusted Computing technology, preventing software corruption necessitates a Trusted Operating System, such as Trusted Gentoo In practice any operating system which aims to be backwards compatible with existing software will not be able to protect against viruses in this way.

Protection of biometric authentication data

Biometric devices used for authentication could use trusted computing technologies (memory curtaining, secure I/O) to assure the user that no spyware installed on his/her PC is able to steal sensitive biometric data. The theft of this data could be extremely harmful to the user because while a user can change a password if he or she knows that the password is no longer secure, a user cannot change the data generated by a biometric device.

2.5 Critism of Trusted Computing

2.5.1 Overview

Trusted Computing opponents such as the Electronic Frontier Foundation and Free Software Foundation claim trust in the underlying companies is not deserved and that the technology puts too much power and control into the hands of those who design systems and software. They also believe that it may cause consumers to lose anonymity in their online interactions, as well as mandating technologies Trusted Computing opponents deem unnecessary. They suggest Trusted Computing as a possible enabler for future versions of mandatory access control, copy protection, and digital rights management.
There is concern amongst critics that it will not always be possible to examine the hardware components on which Trusted Computing relies, the Trusted Platform Module, which is the ultimate hardware system where the core 'root' of trust in the platform has to lie. If not implemented correctly, it presents a security risk to overall platform integrity and protected data. The specifications, as published by the Trusted Computing Group, are open and are available for anyone to review. However, the final implementations by commercial vendors will not necessarily be subjected to the same review process. In addition, the world of cryptography can often move quickly, and that hardware implementations of algorithms might create an inadvertent obsolescence. Trusting networked computers to controlling authorities rather than to individuals may create digital imprimaturs.

2.5.2 Risks

Digital Rigths Management

One of the early motivations behind trusted computing was a desire by media and software corporations for stricter digital rights management technology to prevent users from freely sharing and using potentially copyrighted or private files without explicit permission. Microsoft has announced a DRM technology that says it will make use of hardware encryption.
An example could be downloading a music file from a band: the band's record company could come up with rules for how the band's music can be used. For example, they might want the user to play the file only three times a day without paying additional money. Also, they could use remote attestation to only send their music to a music player that enforces their rules: sealed storage would prevent the user from opening the file with another player that did not enforce the restrictions. Memory curtaining would prevent the user from making an unrestricted copy of the file while it is playing, and secure output would prevent capturing what is sent to the sound system.
Some could say it is far-fetched, but Apple did attach a DRM with all the media files available on i-Tunes (currently this barrier has been lifted for music files, not yet for videos).

Users unable to change software

A user who wanted to switch to a competing program might find that it would be impossible for that new program to read old data, as the information would be ”locked in” to the old program. It could also make it impossible for the user to read or modify their data except as specifically permitted by the software.
Remote attestation could cause other problems. Currently web sites can be visited using a number of web browsers, though certain websites may be formatted such that some browsers cannot decipher their code. Some browsers have found a way to get around that problem by emulating other browsers. With remote attestation a website could check the internet browser being used and refuse to display on any browser other than the specified one (like Internet Explorer), so even emulating the browser would not work.

Users unable to change software

A user who wanted to switch to a competing program might find that it would be impossible for that new program to read old data, as the information would be ”locked in” to the old program. It could also make it impossible for the user to read or modify their data except as specifically permitted by the software.
Remote attestation could cause other problems. Currently web sites can be visited using a number of web browsers, though certain websites may be formatted such that some browsers cannot decipher their code. Some browsers have found a way to get around that problem by emulating other browsers. With remote attestation a website could check the internet browser being used and refuse to display on any browser other than the specified one (like Internet Explorer), so even emulating the browser would not work.

Users have no control over data

Sealed storage could prevent users from moving sealed files to the new computer. This limitation might exist either through poor software design or deliberate limitations placed by publishers of works. The migration section of the TPM specification requires that it be impossible to move certain kinds of files except to a computer with the identical make and model of security chip.

Loss of anonymity

Because a Trusted Computing equipped computer is able to uniquely attest to its own identity, it will be possible for vendors and others who possess the ability to use the attestation feature to zero in on the identity of the user of TC-enabled software with a high degree of certainty.
Such a capability is contingent on the reasonable chance that the user at some time provides user-identifying information, whether voluntarily or indirectly. One common way

that information can be obtained and linked is when a user registers a computer just after purchase. Another common way is when a user provides identifying information to the website of an affiliate of the vendor.
In response to privacy concerns, researchers developed direct anonymous attestation (DAA) which allows a client to perform attestation while limiting the amount of identifying information that is provided to the verifier. DAA also supports an anonymity revocation system wherein a third party has the information necessary to uniquely identify the TPM associated with a particular attestation.

Trust

When you create a private/public key today it is done on the local computer and the creator has complete control over who has access to it. Encryption and decryption chips have a completely static private/public key that is etched into the hardware when it is manufactured, and hardware manufactures have every opportunity to see it and copy it without leaving evidence of doing so. With this key it would be possible to have access to data encrypted with it, and to authenticate as it. It would be fairly trivial manufactures to give a copy of this key to the government or the software manufactures, as the platform must go through steps so that it works with authenticated software. In order to trust anything that is authenticated by or encrypted by a TPM or a Trusted computer, therefore, one has to trust the company that made that chip, the company that designed the chip, those companies allowed to make software for the chip, and the ability and interest of those companies to not compromise the process. It requires a great deal of trust, to have any trust, in Trusted Computing.

2.6 Conclusion

A growing number of profit-minded perpetrators and increasingly sophisticated attacks speak for themselves as threats, and the proliferation of wired and wireless networking and the devices connected to them speaks to the growing threat matrix.
Comprehensive security requires solutions at the user level, the system level, and the network level. The approach of the Trusted Computing Group is the first to attempt a comprehensive, platform-based security solution that ensures privacy, data integrity, and user authentication at all these levels. Microsoft has already launched an application using the TPM specifications and in the coming years TPM will be in all our computers.
However the TPM is still far from perfect. First of all on its design, we can see the example of Microsoft's BitLocker which has been pretty simply hacked. And secondly on a much more moral level, many still fear the power of manufacturers and software companies on the TPM's design. Several groups and individuals have rightly presented the risks of Trusted Computing, often calling it "Treacherous Computing", and fight for a more open implementation.
To conclude, TCG is on the right path, but has yet to arrive to a good equilibrium and we should always keep this quotation in mind:

"They who can give up essential liberty to obtain a little temporary safey, deserve neither liberty nor safety"
- Benjamin Franklin

Literaturverzeichnis

[1] *http://www.bsi.de/sichere_plattformen/trustcomp/infos/tcgi.htm*

[2] *http://www.bsi.de/sichere_plattformen/trustcomp/infos/tpm_report/quellen.htm*

[3] *https://www.trustedcomputinggroup.org/home*

[4] *https://www.trustedcomputinggroup.org/specs/TPM/*

[5] *http://en.wikipedia.org/wiki/Trusted_Computing*

[6] *http://en.wikipedia.org/wiki/Trusted_Platform_Module*

[7] *http://en.wikipedia.org/wiki/BitLocker_Drive_Encryption*

[8] *https://www.trustedcomputinggroup.org/news/Industry_Data/IDC_448_Web.pdf*

[9] *http://www.microsoft.com/whdc/system/platform/hwsecurity/default.mspx*

[10] *http://www.microsoft.com/resources/ngscb/default.mspx*

[11] *http://www.cl.cam.ac.uk/rja14/tcpa-faq.html - FAQ about Trusted Computing by Ross Anderson*

[12] *http://www.eff.org/wp/trusted-computing-promise-and-risk - Electronic Frontier Foundation*

[13] *http://www.opentc.net/index.php?option=com_content&task=view&id=29&Itemid=45 - Open Trusted Computing*

[14] *http://www.gentoo.org/news/20050202-trustedgentoo.xml - Trusted Gentoo*

[15] *http://www.trust.rub.de/home/concluded-projects/trustedgrub/ - TrustedGRUB*

[16] *http://www.perseus-os.org/content/pages/Objectives.htm - PERSEUS*

[17] http://linux.sys-con.com/node/47423 - Article on Trusted Computing in Free/Open Technologies

[18] *http://enforcer.sourceforge.net/ - Enforcer*

[19] *http://www.gnu.org/philosophy/can-you-trust.html*

[20] *http://www.lafkon.net/tc/*

[21] *http://citp.princeton.edu/memory/*

[22] *http://fr.wikipedia.org/wiki/Next-generation_secure_computing_base*

[23] *http://fr.wikipedia.org/wiki/Microsoft_Windows_Vista*

[24] *http://www.april.org/groupes/informatique-deloyale/*

[25] *http://www.zdnet.fr/actualites/informatique/0,39040745,2133649,00.htm*

Kapitel 3

SINA - Sichere Inter-Netze Architektur

Robert Koch

Im folgenden Kapitel wird ein Überblick über die SINA-Architektur gegeben. SINA steht für Sichere Inter-Netze Architektur und ist ein Produkt der secunet Security Networks AG[1].

[1] www.secunet.de

Inhaltsverzeichnis

3.1 Einführung

3.1.1 Ursprung und Motivation

Nach der deutschen Wiedervereinigung entbrannte schnell die Hauptstadtdebatte, die leidenschaftlich von den politischen Lagern geführt wurde. Mit denkbar knapper Mehrheit setzte sich schließlich Berlin als künftige Hauptstadt des geeinten Deutschlands durch, mit einem Stimmenverhältnis von 338 zu 320 Stimmen. Der damit verbundene Umzug und daraus resultierende Bedarf einer sicheren Kommunikation zwischen Berlin und Bonn, aber auch die Erforderniss einer modernen, vernetzten Kommunikation waren treibende Kräfte zur Initiierung des *SINA*-Projekts. Auch die bis dato aufwändigen Verfahren im Bereich des Geheimschutz waren nicht mehr zeitgemäß, die strengen Richtlinien zum Umgang mit eingestuftem Material, Datentransport per Kurierwesen und weitere Anforderungen schränkten die Arbeitsfähigkeit stark ein.
Die Anforderungen an eine moderne Kommunikation liegen hier nicht nur im sicheren Anbinden von Außenstellen über nicht- vertrauenswürdige Netze, wie beispielsweise die Vernetzung der über 200 Botschaften und Auslandsvertretungen von Deutschland, sondern auch in der Verbindung von eingestuften Teilnetzen über unsichere Kommunikationsmittel.

3.1.2 Begriffsdefinition

Nachfolgend werden die im weiteren Verlauf benötigten Begrifflichkeiten definiert. Zunächst werden die Betriebsarten der Informationsverarbeitung dargestellt. Es kann festgehalten werden, dass die meisten Netze einem einfachen Sicherheitsmodell unterliegen: Es erfolgt lediglich eine Trennung zwischen dem inneren, eingestuften bzw. restriktiven Netz, und dem äußeren Netz. Das innere Netz wird als sicher angenommen, das äußere als nicht- vertrauenswürdig. Die einfachste Erweiterung dieses Konzepts, zum Beispiel im Rahmen der Netze verschiedener Firmenniederlassungen, ist die Verbindung zweier innerer Netze über ein drittes Netz, mittels einer gesicherten Verbindung *(Virtual Private Network, VPN).*

System High

Ein *System High* ist die einfachste Art, Informationen in einem Netz zu verarbeiten. Hier werden alle Daten von *OFFEN* bis einschließlich der Einstufung des Netzes gemeinsam gespeichert und verarbeitet. Dies ermöglicht eine einfache Infrastruktur, bedingt aber insbesondere, dass alle im Netz vorliegenden Daten mit der höchsten Klassifizierung, für die das Netz freigegeben ist, eingestuft werden müssen. Da dies im gleichen Zug erfordert, dass jeder Nutzer über die Ermächtigung zum Zugang zur entsprechenden Sicherheitsstufe verfügen muss, kann das *need-to-know* Prinzip hier nicht mehr gehalten werden. *Need-to-know*, bzw. in den Streitkräften als *Wissen-nur-wenn-nötig* bekannt, bezeichnet

die Regelung, dass Zugang zu Information bei Vorhandensein der entsprechenden Ermächtigung auch nur dann gewährt wird, wenn dies aus dienstlichen bzw. arbeitstechnischen Gründen erforderlich ist.

Dedicated

In einem *Dedicated* System haben alle Daten die gleiche Einstufung. Die Auswahl der Nutzer erfolgt so, dass für den jeweiligen Bereich der gleiche *need-to-know* Umfang vorliegt. Entsprechend haben alle Nutzer auch die gleiche Sicherheitsermächtigung. Dies ermöglicht eine deutlich bessere Granularität des Zugriffs, bedingt gegebenenfalls jedoch umfangreichere Maßnahmen bzgl. der Infrastruktur, wenn mehrere Bereiche verschiedener Einstufung vorgehalten werden müssen.

Multi-Level-Secure

Um ein flexibles Arbeiten zu gewährleisten, und sowohl verschiedene Sicherheitsstufen von Daten zu ermöglichen, als auch den unterschiedlichen Sicherheitsermächtigungen Rechnung zu tragen, kann eine *Multi-Level-Secure (MLS)* Umgebung geschaffen werden. Hier stellt ein zertifizierter Mechanismus sicher, dass sowohl

- jedes Objekt[2] vertrauenswürdig eingestuft ist,
- jeder Anwender und jedes Programm einen bestimmten Ermächtigungslevel besitzen und
- ein Zugriff nur erfolgen kann, wenn der Ermächtigungslevel für die angeforderten Daten ausreichend ist.

Dieses Verfahren bietet ein hohes Maß an Flexibilität, ist jedoch sehr aufwändig in der Umsetzung. Insbesondere müssen die sichere Erstellung und Prüfung der Kennzeichnungen[3] gewährleistet und die Entscheidungsmethode sicher sein.
Bei großen Systemen ist daher oftmals die Umsetzung in Form von separaten Teilnetzen unterschiedlicher Klassifizierung als *System High* oder *Dedicated* Netze günstiger, als die Realisierung eines *MLS* Systems.

Compartmented

Als Zwischenstufe von *System High* und *MLS* Systemen haben sich die sog. *Compartmented* Systeme gebildet. Hierbei werden Mechanismen zur grobgranularen Trennung von Daten zur Verfügung gestellt. Dies ist auch das Entwicklungskriterium, das für die *SINA-Client-* Plattform genutzt wird.

[2] Dateien, Verzeichnisse, ...
[3] *Labels*

3.1.3 Arbeitsprinzipien

Um die Voraussetzungen für den Umgang mit eingestuften Daten zu beleuchten, und die damit resultierenden Anforderungen an die verarbeitende Plattform zu entwickeln, werden im folgenden die notwendigen Arbeitsprinzipien eingeführt.

Datenaustausch

Grundanforderung der Datenverarbeitung ist der wachsende Bedarf, auch eingestufte Daten kontrolliert auszutauschen, ohne dafür große *System High* oder *Dedicated* Netze zu bilden.

Sanitisation

Mit *Sanitisation* wird eine *Gateway-* Funktion bezeichnet. Hierbei handelt es sich um einen *Reinigungs-* Prozess, der alle den Gateway passierende Dokumente untersucht, und jegliche Informationen entfernt, welche die Einstufung des Zielnetzes überschreiten.

Labelling

Wurde eine *Sanitisation* des Dokumentes durchgeführt, befinden sich nur noch Daten mit der maximalen Einstufung des Zielnetzes darin. Um sicherzustellen, dass nach diesem Vorgang nicht erneut höher eingestufte Informationen in das Objekt gelangen, muss dieses seiner Einstufung entsprechend gekennzeichnet werden. Dies erfolgt durch entsprechende Ergänzung der digitalen Signatur. Um die Wirksamkeit zu gewährleisten, muss dieser Vorgang, der als *Labelling* bezeichnet wird, unmittelbar nach der *Reinigung* durchgeführt werden.

Registry

Der Prozess der *Registrierung* dient dem lückenlosen Nachweis über die Exemplare bzw. Kopien höher als *VS-NfD* eingestufter Dokumente und deren Verbleib. Er schließt sich an das *Labelling* an. Der gesamte Ablauf – *Sanitisation*, *Labelling* und *Registration* – wird als *sicherer Workflow* bezeichnet.

3.1.4 Anforderungen an die Architektur

Aus den genannten Voraussetzungen für eine sichere Verarbeitung vertraulicher Daten lassen sich eine Reihe von Anforderungen an das System ableiten:

- **Vertrauenswürdige Plattform:** Dies bedingt ein umfassendes Verständnis der Abläufe und Prozesse des Betriebssystems und erfordert somit die Möglichkeit, die Codebasis einzusehen. Darüberhinaus ist ein modularer Aufbau erforderlich, um ohne großen Mehraufwand eine möglichst kleine Systembasis zu bilden. Skalierbarkeit und Kostenaspekte müssen ebenfalls bei den Plattformaspekten berücksichtigt werden.

- **IPsec-basiertes VPN-Subsystem:** Die Grundlage der verschlüsselten Kommunikation soll das Sicherheitsprotokoll *IPsec* liefern. Dessen Architektur wird maßgeblich in den `RFCs 2401` und `4301` beschrieben. Es arbeitet auf Layer 3 des *OSI-Referenzmodells* und definiert Security Services für IPv4 sowie IPv6. Die definierten Dienste reichen von der Datenverschlüsselung über Zugangskontrolle bis zu drahtloser Sicherheit.

- **Sicher abgeschottete Bereiche:** Von besonderer Bedeutung ist es, dem Nutzer die Möglichkeit zur Verfügung zu stellen, verschiedenartig klassifizierte Dokumente auf dem selben System zu bearbeiten. Hierfür müssen abgeschottete Sitzungen möglich sein, die parallel verschieden eingestufte Arbeitsumgebungen anbieten können.

- **Kryptographisches Dateisystem:** Müssen eingestufte Informationen gespeichert werden, ist die Ablage in verschlüsselter und entsprechend der Klassifizierung freigegebenen Form zu gewährleisten. Besonderes Augenmerk muss darauf gelegt werden, dass keine Daten in nicht-permanenten Speicherbereichen verbleiben, dies betrifft maßgeblich den *SWAP*[4]- Bereich, sowie die in temporären Verzeichnissen abgelegten Dateien[5].

- **Konfigurierbarer Zugang zu lokalen I/O- Schnittstellen:** Die physikalischen Zugriffsmöglichkeiten bedürfen einer besonderen Beachtung. Ein nicht-autorisiertes Abfliessen von eingestuften Daten über vorhandene Schnittstellen darf nicht möglich sein. Alle Zugangs- und Datenübertragungsmöglichkeiten mittels lokaler Schnittstellen müssen abhängig der Einstufung sicher konfigurierbar sein.

- **Mehrere Bereiche unterschiedlicher Sicherheitsstufen auf einer Plattform:** Der bisherigen Anforderung, für unterschiedlich eingestufte Daten entsprechend mehrere getrennte Systeme vorzuhalten, soll durch die Einführung paralleler, gekapselter Umgebungen Rechnung getragen werden. Somit wird die Bearbeitung unterschiedlich eingestufter Daten in getrennten, völlig isolierten Sessions auf einer physikalischen Einheit möglich.

- **Konfiguration streng kontrollierter Übergänge:** Jegliche Netzverbindungen, insbesondere auch zwischen unterschiedlich eingestuften Netzen, müssen sicher und administrativ einfach eingerichtet werden können. Hierbei ist von essentieller Bedeutung, dass die VPN-Schicht nicht umgangen werden kann.

Allgemein muss die Plattform flexibel und skalierbar sein, modular aufgebaut und kostengünstig umgesetzt werden können. Wo möglich, sind *COTS*-[6]Produkte einzusetzen.

[4] Auslagerungsbereich des Arbeitsspeichers.
[5] z.B. `/tmp`
[6] Commercial of the Shelf

3.1.5 Grundlagen der *SINA*-Architektur

Um den Anforderungen aus 3.1.4 gerecht zu werden, sind folgende Komponenten für die Entwicklung einer sicheren Kommunikationsplattform ausgewählt:

Plattform

Basis für die *SINA*-Plattform bildet ein GNU-Linux Kernel der Version `2.6.12`. Durch den als *Open Source* verfügbaren Quellcode lassen sich alle Vorgänge des Betriebssystems nachvollziehen und prüfen, eine minimale, der Architektur entsprechend angepasste Code Basis erleichtert dies. Weitere wichtige, unter Linux umgesetzte Punkte sind:

- Page Execution Protection Subsystem
- Rollenbasierte Zugangskontrolle
- Kryptographisches Dateisystem
- Virtualisierungsschicht

Neben diesen Punkten ist es notwendig, dem Anwender eine *"look-and-feel"* Umgebung zur Verfügung zu stellen. Die gewählte Architektur bietet durch die Möglichkeit der Integration von X Version 11[7] und der Unterstützung zahlreicher Remote-Protokolle umfangreiche und bekannte graphische Arbeitsumgebungen.

Der Aspekt der Mobilität kann durch das ausgewählte Betriebssystem ebenfalls in den entsprechenden Punkten - Hardwareunterstützung, Netz- sowie kryptographische Komponenten erfüllt werden.

Kryptographische Primitive

Die sichere Verschlüsselung ist elementare Grundlage der Architektur. Um alle Bereiche mit den unterschiedlichen Anforderungen an Sicherheit, Zertifizierung, etc. abzudecken, sind sowohl symmetrische als auch asymmetrische Algorithmen zu integrieren, wiederrum als Software als auch in Form von Krypto-Hardware für den Hochsicherheitsbereich. Zuverlässige Zufallszahlengeneratoren sowie die Möglichkeit der Nutzung von *SmartCards* sind ebenso vorzusehen.
Im Rahmen der Netzsicherheit wird der Anforderung entsprechend *IPsec* als Basis eingesetzt. Grundlegende Filterfunktionen werden mittels *iptables* umgesetzt, ebenfalls werden durch den Linux-Kernel Paketmarkierung und -verfolgung ermöglicht. Für die Handhabung und Verwaltung des Kryptomaterials sind das *IKE*-[8] Protokoll für das Schlüsselmanagement und ein Zertifikatsmanagement vorzusehen.

[7] www.x.org
[8] Internet Key Exchange

Sessionorientierte Funktionen

Um das Konzept, mehrere unterschiedlich eingestufte Sitzungen auf einem Rechner parallel ausführen zu können umzusetzen, sind sogenannte *Sessions* erforderlich. Diese bilden die Schnittstelle zum Anwender mit einer bestimmten Sicherheitsstufe. Die Nutzung von *SINA*-Linux ermöglicht die Ausführung von bis zu sechs unterschiedlich klassifizierten Sessions parallel.
Die Session-Typen müssen ein umfangreiches Feld abdecken, von der Verbindung eines *Remote Desktops* bis zur Nutzung von Netzspeicher oder Telekommunikationsdiensten. Im einzelnen stehen hier derzeit folgende Sessions zur Verfügung:

- RDP, X11, ICA
- Virtual Machine
- iSCSI
- VoIP

Weiterhin befinden sich derzeit in Entwicklung:

- Quarantäne und
- Transfer-Session.

Diese erlauben dem Nutzer, nicht vertrauenswürdige Dokumente in einer abgesicherten Umgebung zu nutzen, bzw. die gesicherte Übertragung von eingestuften Informationen gemäß den Policies.

Weitere Komponenten

Ergänzende Funktionalitäten werden weiterhin in den Bereichen *Manipulationsschutz*, *Ressourcenmanagement* sowie *Intrusion Detection and Response* benötigt.

Architektur SINA-Box

Abbildung 3.1 zeigt schematisch den Aufbau der *SINA*-Box.

Basis der *SINA*-Architektur sind Standard-Hardware-Komponenten (*COTS*). Dies ermöglicht einen kostengünstigen Aufbau des Systems. Jede *SINA*-Box ist ein *IP Krypto Gateway* nach *IPsec* Standard. In der Ausführung als *SINA*-Box sind keine beschreibbaren, permanenten Speicher vorgesehen. Je nach Ausstattung verfügt die Box über jeweils ein bis vier schwarze und rote Netzinterface-Karten. Das minimale, gehärtete Betriebssystem wird von einer *CD-ROM* oder einem *Flash*-Speicher gestartet.

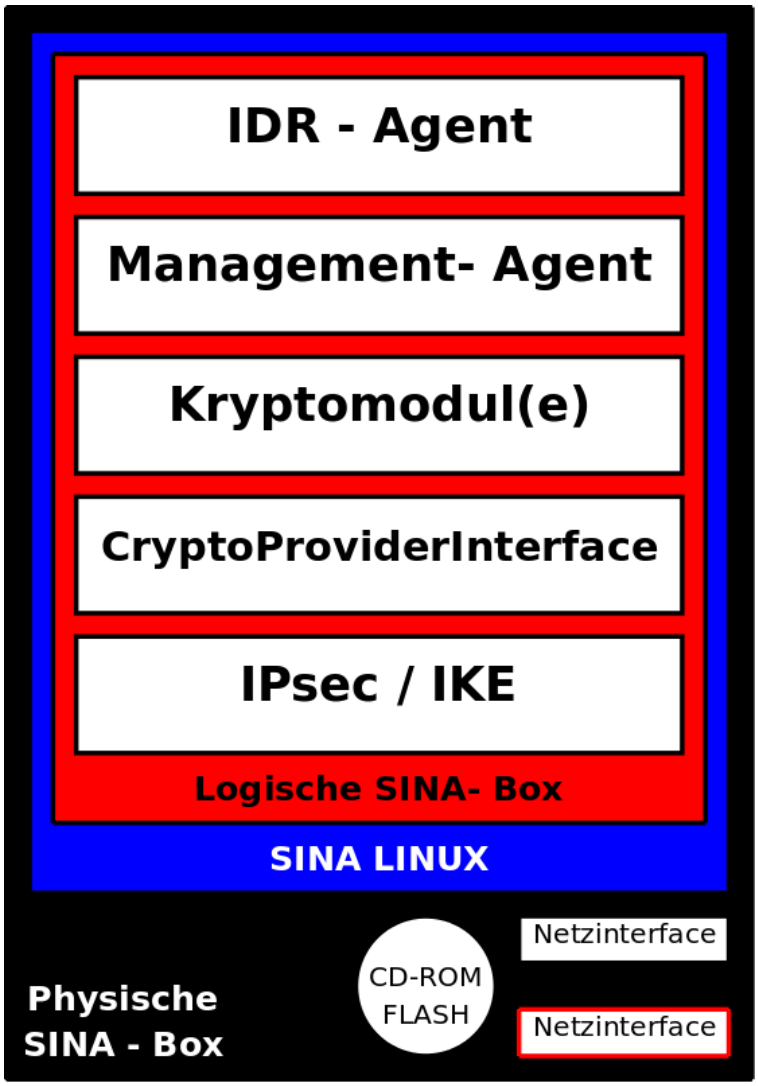


Abbildung 3.1: Architektur der SINA-Box[3]

Die Authentifizierung, sowie grundlegende Initialisierung des Systems erfolgt mittels einer *SmartCard*, welche über das *CryptoProviderInterface* angebunden ist. Diese stellt weiterhin einen Hardware-Rauschgenerator zu Erzeugung sicherer Zufallszahlen zur Verfügung.

Abhängig der Einstufung der zu verarbeitenden Daten bzw. deren Anforderungen, ist die Box in Tamper sowie Tempest- Varianten lieferbar.

Tamper bezeichnet den Schutz eines Gerätes vor Sabotage. Im Bereich der Sicherheit von Kryptomaterial sind hier insbesondere Maßnahmen zu verstehen, die auf eine Vernichtung der auswertbaren Daten, Algorithmen, Schaltkreise, etc abzielen, um so eine Auswertung zu verhindern. Beispielsweise können hier Kontakte am Gehäuse angebracht werden, welche beim Öffnen des Gerätes vorhandene Schlüssel aus den Speichern löschen (vgl. Abbildung 3.2).

Tempest ist ein Ende der 60er Jahre durch die US-Regierung eingeführtes Codeword. Tempest hat zum Ziel, elektronisches Gerät vor kompromittierender Abstrahlung zu schützen,

um somit ein Abhören der elektromagnetischen Ausstrahlung zu verhindern. Auch als *Emission Security* oder *EMSEC* bezeichnet.

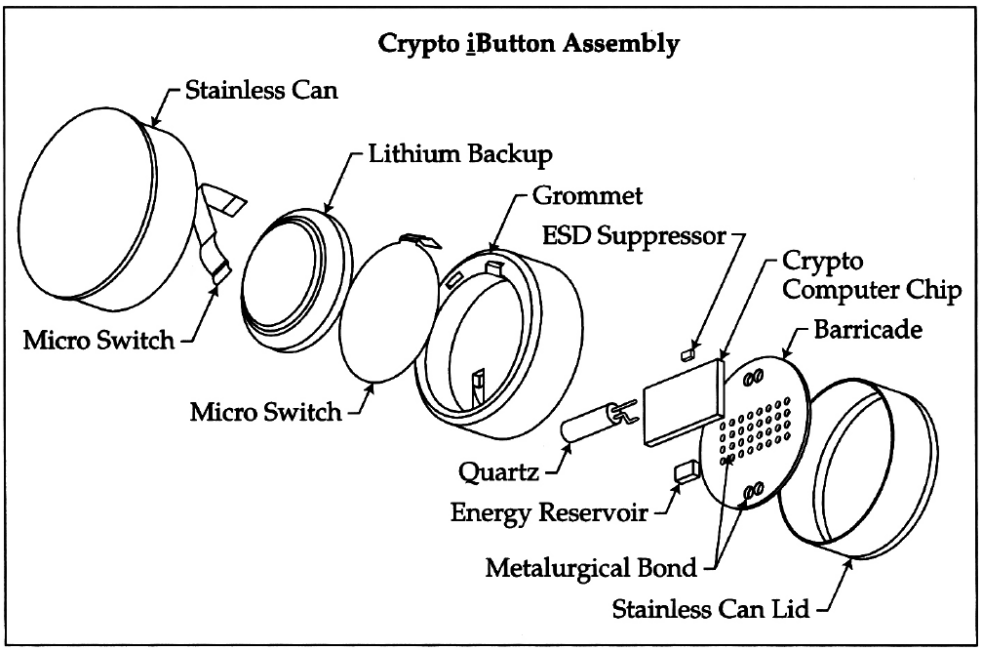


Abbildung 3.2: Beispiel eines Medium-Security-Processor: iBotton von Dallas Semiconductor Inc. Kontaktschalter zum Erkennen des Öffnens des Gehäuses[4]

Die Box ist in den Varianten *S* sowie *H/P* lieferbar. Erstere stellt eine auf Software-Algorithmen bzw. Modulen umgesetzte Variante dar, während die *H/P*-Modelle mit Krypto-Hardware ausgestattet sind.

Architektur SINA-ThinClient

Der *SINA*-ThinClient ist gemäß Abbildung 3.3 aufgebaut.

Der prinzipielle Aufbau der Hardware ist entsprechend dem der *SINA*-Boxen. Auch beim ThinClient ist keine Festplatte vorhanden. Sämtliche Daten werden über das Netz mittels einer Remote-Session zur Verfügung gestellt. Da der Betrieb des ThinClients in einer nicht-vertrauenswürdigen Umgebung abläuft, wird die gesamte Kommunikation verschlüsselt über das schwarze Interface abgewickelt. Auf ein rotes Netzinterface kann verzichtet werden, jedoch verfügt der ThinClient über eine Grafikkarte zur Anzeige der Arbeitsumgebung, welche in den *SINA*-Boxen nicht notwendig ist. Alle Bildschirminhalte werden verschlüsselt über einen *IPsec*-Tunnel übermittelt, und lokal auf dem X-Server dargestellt. Wie bereits erwähnt, können mehrere Sessions unterschiedlicher Einstufungen parallel ausgeführt werden. Diese laufen als gekapselte Umgebungen ab, und befinden sich dabei *innerhalb* der logischen *SINA*-Box. Dies ermöglicht ein hohes Maß an Sicherheit und die erforderliche Trennung.

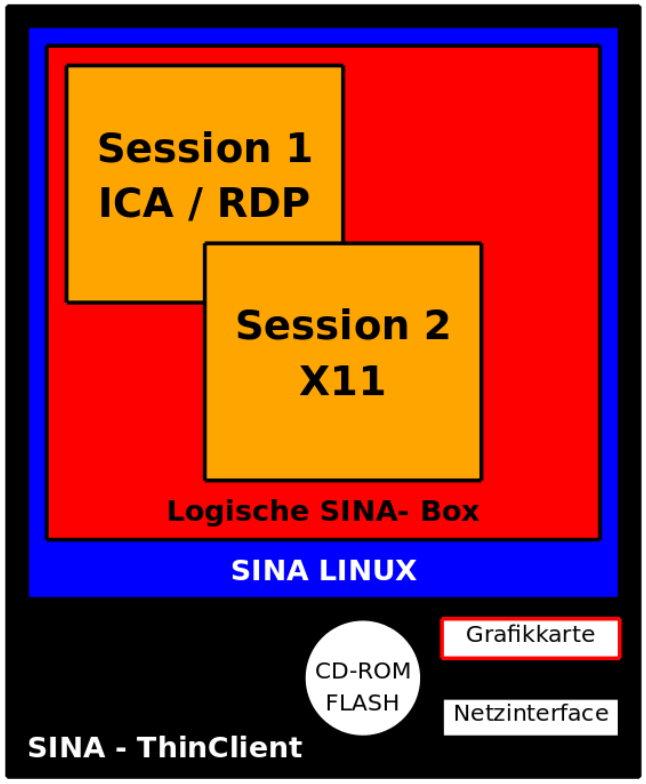


Abbildung 3.3: Architektur des SINA-ThinClient[3]

3.2 SINA-Komponenten

Der nachfolgende Abschnitt gibt einen Überblick der verfügbaren *SINA*-Komponenten. An diese kurze Einführung schliessen sich in Kapitel 3.3 die Einsatzmöglichkeiten an.

3.2.1 Infrastruktur-Komponenten

Im Bereich der Infrastruktur-Komponenten stehen die *SINA*-Boxen und -Gateways zur Verfügung, mit deren Hilfe verschlüsselte Kommunikationsstrecken über unsichere Netze aufgebaut werden können.

SINA-Box

Die eigentliche *SINA*-Box ist in Abbildung 3.4[9] ersichtlich.

[9] Alle Photos sind den Datenblättern gemäß [3] entnommen.

Abbildung 3.4: Ansicht der *SINA*-Box

Gut erkennbar ist die Nutzung von Standard-Hardwarekomponenten. Darüber hinaus sind der Slot für die SmartCard, sowie das Eingabefeld für den Code erkennbar. Als Gehäuse können neben üblichen 19"-Einschüben auch solche geliefert werden, die den Kriterien für Tempest und Temper entsprechen. Die Boxen stehen in den Varianten *H/P* sowie *S* bereit. Bei *H/P* erfolgt die Verschlüsselung mittels Krypto-Hardware, hierfür wird die Kryptokarte *Pluto* mit dem Algorithmus *Libelle* eingesetzt, welche bis zur Einstufung *Streng Geheim* zugelassen ist. Die Software-Algorithmen (*S*) sind bis *VS-Vertraulich* freigegeben.

Layer2-Gateway

Als Ergänzung zu den *SINA*-Boxen steht ein *Layer2*-Gateway zur Verfügung (vgl. Abbildung 3.5).

Abbildung 3.5: Ansicht des *SINA*-Layer2-Gateways

Dieser ermöglicht eine transparente Verschlüsselung direkt auf dem Layer2. Somit lässt sich der Gateway in einer bestehenden Infrastruktur einsetzen, ohne dass diese geändert oder angepasst werden muss. Der Gateway ermöglicht Punkt-zu-Punkt-Verschlüsselung und ist somit eine Ergänzung zu den *SINA*-Boxen. Derzeit verfügbare Varianten sind mit einem Durchsatz von bis zu 10 Gbps erhältlich.

3.2.2 Endgeräte

Drei Varianten der für den Anwender verfügbaren Endgeräte stehen bereit.

SINA-ThinClient

Der Aufbau des ThinClients wurde in Kapitel 3.1.5 vorgestellt. Merkmal ist insbesondere, dass dieser als Diskless-System umgesetzt ist, eine lokale Speicherung von Daten erfolgt

nicht. Für die Nutzung muss eine Online-Verbindungen zu einem geschützten Terminal-Server über eine *SINA*-Box aufgebaut werden. Sämtlicher Datentransfer wird mittels der Remote-Protokolle *RDP*, *X11* und *ICA* abgewickelt. Hierbei werden maßgeblich Bildschirmseiten übertragen. Abbildung 3.6(a) zeigt einen ThinClient.

SINA-Virtual Workstation

Die *SINA*-Virtual Workstation basiert auf der OEM-Variante von *VMware*[10]. Dieses stellt die Kapselumgebung für ein komplettes, nicht vertrauenswürdiges Betriebssystem dar. Hierbei bildet der Virtual Machine Monitor die Schnittstelle zur Hardware. Im Gegensatz zum ThinClient ist hier eine Offline-Bearbeitung der Daten möglich, die ebenfalls lokal verschlüsselt abgelegt werden können. Abbildung 3.6(b) zeigt die Virtual Workstation.

(a) Ansicht des *SINA*-ThinClient

(b) Ansicht der *SINA*-Virtual Workstation

(c) Ansicht der *SINA*-Virtual Desktop

Abbildung 3.6: *SINA*-Endgeräte

[10] www.vmware.com

***SINA*-Virtual Desktop**

Um die durch *SINA* gebotene sichere Kommunikation und Speicherung von Daten auch einfach in bereits bestehende Umgebungen integrieren zu können, wird der *SINA*-Virtual Desktop bereitgestellt. Dieser lässt sich auf einem bestehenden Windows-Arbeitsplatz installieren und kapselt die sensiblen Daten innerhalb der nicht-vertrauenswürdigen Windows-Umgebung (vgl. Abbildung 3.6(c)).

3.3 Einsatz-Szenarien

Im folgenden werden unterschiedliche Szenarien vorgestellt, anhand derer die Einsatzmöglichkeiten der *SINA*-Komponenten näher beleuchtet werden.

3.3.1 Basis-Szenarien

Abbildung 3.7 zeigt das elementare Anwendungsszenario. Hierbei wird der Zugang eines in einer nicht-vertrauenswürdigen Umgebung stehenden ThinClients zu einem Terminal-Server im klassifizierten Netz ermöglicht. Die über das unsichere Netz laufende Kommunikation wird dabei vom ThinClient bis zur an der Grenze des klassifizierten Netzes befindlichen *SINA*-Box verschlüsselt. Innerhalb des klassifizierten Netz können die Daten unverschlüsselt zum TerminalServer weitergeleitet werden. Maßgeblicher Vorteil ist, dass die hohen Schutzanforderungen auf einen kleinen Bereich reduziert werden können, was sowohl die Sicherheit erhöht und die Administration vereinfacht, als auch eine Kostenersparnis ermöglicht.

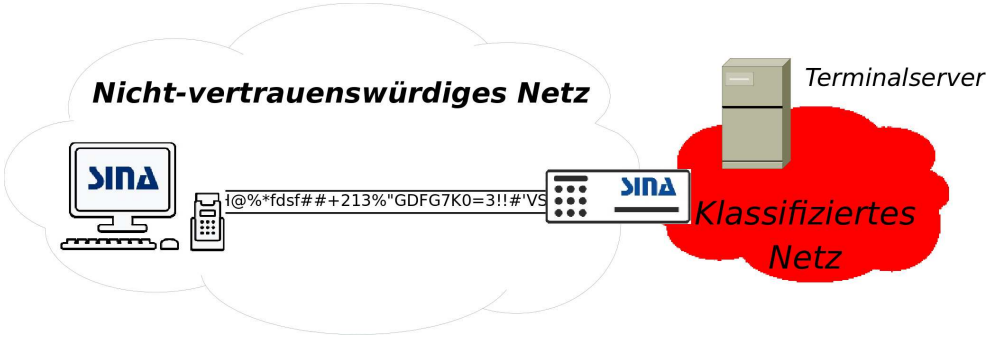


Abbildung 3.7: ThinClient-Access über ein nicht-vertrauenswürdiges Netz

Innerhalb des ThinClients werden keine Daten gespeichert, die Übertragung erfolgt verschlüsselt in Form von Screenshots.

Eine der grundlegenden Forderungen war, dass mehrere Sessions unterschiedlicher Einstufung gleichzeitig auf dem selben System ermöglicht werden. Dies wurde durch gekapselte

Sessions möglich, wovon bis zu sechs gleichzeitig auf einem System ausgeführt werden können. Abbildung 3.8 zeigt den daraus entstehenden, möglichen MultiAccess-Einsatz eines ThinClients.

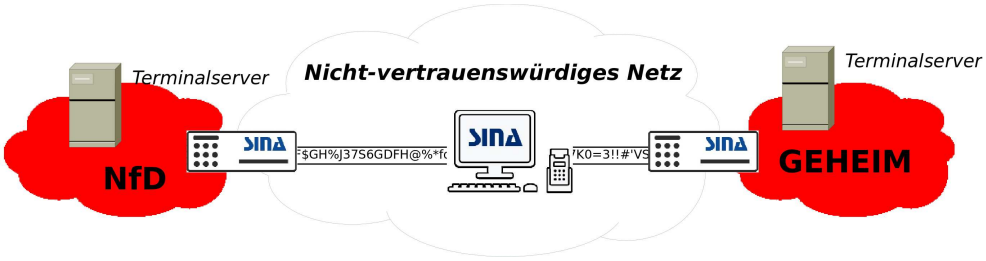


Abbildung 3.8: Nutzung verschiedener Sessions unterschiedlicher Einstufung auf einem ThinClient

Der Vorteil der Multi-Sessions wird hier besonders deutlich. Statt der bisherigen Notwendigkeit, für jeden Schutzbereich ein extra System vorzuhalten, mit welchem nur die jeweiligen Informationen verarbeitet werden dürfen, lassen sich mittels des ThinClients und der Sessions unterschiedliche Einstufungen auf demselben System bearbeiten. Die Kapselung sorgt dabei für die Sicherheit der einzelnen Sessions und deren Abgrenzung untereinander.

Ein typisches Anwendungsszenario zeigt Abbildung 3.9. Oftmals existieren durch Niederlassungen, etc. unterschiedliche Teilnetze, welche miteinander verbunden werden sollen. Da aus Kostengründen in den meisten Fällen keine eigene Verbindungsinfrastruktur aufgebaut werden kann, müssen öffentlich verfügbare Netze als Kommunikationsmedium genutzt werden, insbesondere das Internet.

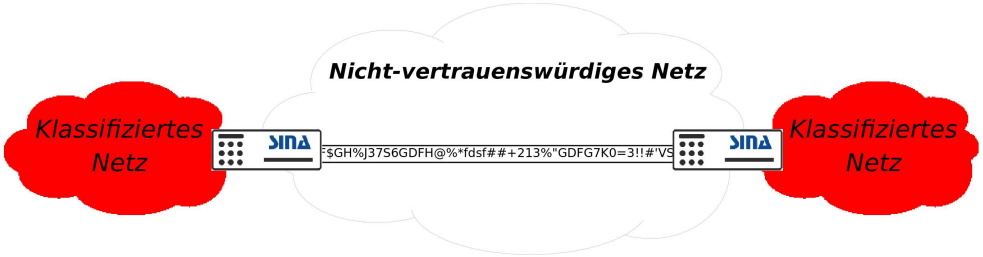


Abbildung 3.9: Verbindung klassifizierter Netze über nicht-vertrauenswürdige Transportnetze

Mittels der *SINA*-Boxen können eingestufte Teilnetze leicht über nicht-vertrauenswürdige Netze verbunden werden. Die Verschlüsselung erfolgt wiederum an den Netzübergängen, innerhalb der klassifizierten Teilnetze sind keine weiteren Maßnahmen erforderlich.

Der Aufbau eines komplexeren Szenarios wird in Abbildung 3.10 dargestellt. Hier werden mehrere *VPN* mittels *SINA*-Boxen verbunden.

Konnektionen sind dabei nicht auf zwei Teilnetze beschränkt, sondern es lassen sich beliebig komplexe Szenarien entwickeln. Neben der Verbindung von Subnetzen findet sich hier

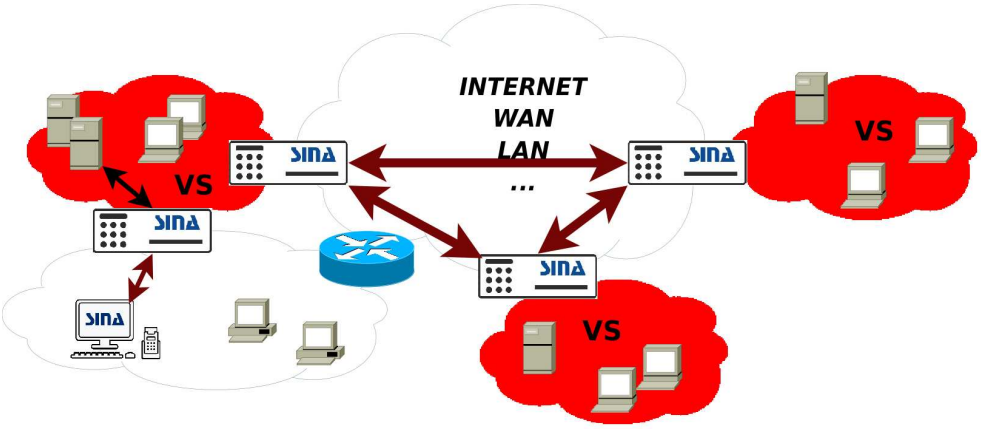


Abbildung 3.10: VPN-Szenario mit *SINA*-Boxen

auch die Einbindung einzelner Hostrechner sowie die Nutzung der ThinClients. Innerhalb der VPN-Struktur können flexible Sicherheitsdomänen etabliert werden.

Eine weitere Einsatzmöglichkeit ist die Nutzung des ThinClients als sogenannter *SINA Inverse ThinClient* (vgl. Abbildung 3.11).

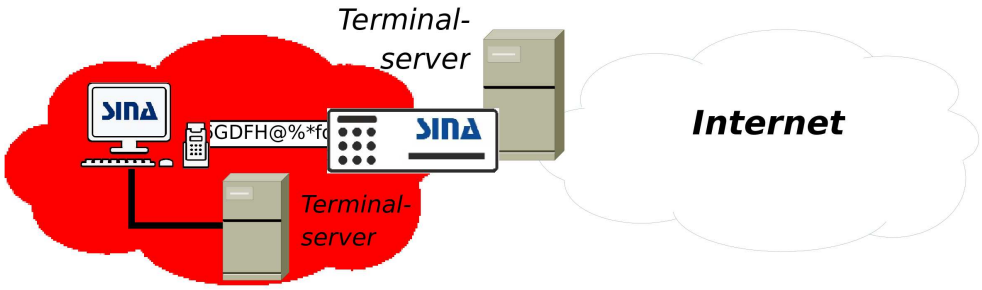


Abbildung 3.11: Einsatz als *Inverse ThinClient*

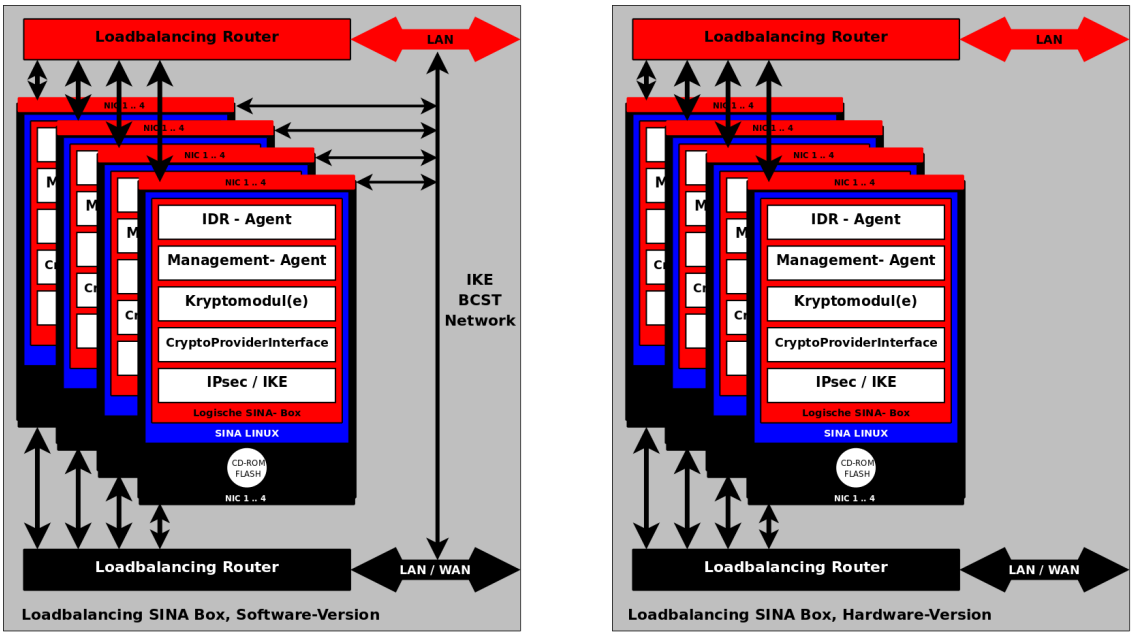
Hierbei wird der Client-Zugriff auf eine nicht-vertrauenswürdige Quelle abgesichert: Hier werden die Daten am Netzübergang mittels einer *SINA*-Box verschlüsselt auf einem gesicherten Kommunikationskanal innerhalb der eingestuften Umgebung übertragen, und an einen ThinClient weitergegeben. Dort sind die Daten innerhalb einer gekapselten Umgebung verfügbar. Somit können die nicht-vertrauenswürdigen Daten nicht in die eigentliche, klassifizierte Umgebung gelangen, das Einbringen von Schadsoftware wird somit unterbunden.

Durch das Session-Konzept ist weiterhin der gleichzeitige Zugriff auf eingestufte Daten mittels der Ausführung paralleler Sitzungen im selben System möglich.

3.3.2 Erweiterte Szenarien

Insbesondere starke kryptografische Verfahren zur Absicherung der Kommunikation fordern eine hohe Rechenleistung. Um den Anforderungen steigender Bandbreiten flexibel

nachzukommen, können die *SINA*-Boxen sowohl in der *H/P*- als auch in der *S*- Variante in einem *LoadBalancing*- Verbund eingesetzt werden. Abbildung 3.12(a) zeigt die Umsetzung in der Software-Variante. Die notwendige Versorgung mit Schlüsseln erfolgt über eine *Public Key Infrastructure* und wird mittels eines Broadcast-Netzes durch das *Internet Key Exchange* Protokoll verwirklicht.



(a) Loadbalancing bei *SINA (S)*-Boxen (b) Loadbalancing bei *SINA (H/P)*-Boxen

Abbildung 3.12: Loadbalancing mittels *SINA*-Boxen

Bei Vorliegen von Hardware-Kryptokarten kann das zusätzliche Netz entfallen. Abbildung 3.12(b) zeigt den entsprechenden Aufbau. Als Kryptokarte kommt die im Projekt *PEPP-1* entwickelte Pluto-PCI-Karte zum Einsatz, deren Verschlüsselung auf dem Libelle-Algorithmus beruht. Mittels der Verteilung der Last auf mehrere *SINA*-Boxen durch *Loadbalancing-Router* kann ein entsprechend höherer Durchsatz erreicht werden. Tabelle 3.1 zeigt einen Überblick der Leistungsfähigkeit der einzelnen *SINA*-Komponenten.

Umsetzung	Algorithmus	max. Bandbreite
Software	AES	50 Mbps
Software	3DES	30 Mbps
Hardware	Libelle	80 Mbps

Tabelle 3.1: Vergleich des maximalen Datendurchsatz verschiedener *SINA*-Box-Konfigurationen auf einem *P-III 866 MHz*

Ein weiteres fortgeschrittenes Einsatzszenario lässt sich mittels des in Entwicklung befindlichen *SINA Security Gateway* verwirklichen.

Die Plattform hierfür stellt eine *SINA Virtual Workstation* dar. Diese wird um Gateway- und serverorientierte Funktionalitäten erweitert. Der physikalische In- und Output ist

streng reglementiert festgelegt. Im Inneren des Security Gateway findet ein stringenter Datenfluss statt, der mittels Filtern, Firewalling-Funktionalität und Datenkonvertierungsfunktionen sicherstellt, dass nur dem Zielnetz entsprechend klassifiziertes Material weitergeleitet wird. Hierfür ist eine Umsetzung der aufwändigen, in Kapitel 3.1.3 angerissenen Funktionen notwendig. Vorteil hierbei ist neben einer hohen Flexibilität der Weiterleitung in verschiedene Zielnetze auch die Möglichkeit, unterschiedlich klassifizierte Informationen in einem Netz zu speichern und zu bearbeiten. Dieses muss entsprechend der Einstufung der am höchsten klassifizierten Daten des Netzes entsprechen, durch die Nutzung des Security Gateway entfällt jedoch die Notwendigkeit, alle im Netz befindlichen Daten entsprechend der Netzfreigabe hochzustufen.

3.3.3 Management

Um eine einfache und sichere Verwaltung und Handhabung der Sicherheitsbeziehungen zu ermöglichen, ist eine entsprechende Management-Funktionalität umgesetzt. Abbildung 3.13 zeigt den logischen Aufbau.

Der grundlegende Ablauf der Administration der *SINA*-Komponenten skizziert sich wie folgt:

1. Wird eine neue Komponente integriert, wird mittels der *Registration Authority* ein entsprechendes Zertifikat angefordert.
2. Die *Certification Authority* erstellt ein entsprechendes Zertifikat, welches in einer rationalen Datenbank abgelegt wird. Weiterhin wird eine SmartCard mit der notwendigen Krypto-ID ausgegeben.
3. Mittels des *Configuration and Access Control List Manager* können die notwendigen Konfigurationen der neuen Komponente vorgenommen werden; diese werden in der Datenbank abgelegt, und sind mittels eines Verzeichnisdienstes zu erreichen.
4. Wird die neue Komponente mittels der SmartCard gestartet, erfolgt hiermit eine Authentifizierung sowie eine initiale Konfiguration. Weitere Konfigurationskomponenten werden nach dem Boot-Vorgang mittels des Verzeichnisdienstes aus der Datenbank ausgelesen. Wichtig hierbei ist, dass **kein** interaktives Management der Komponenten möglich ist, was ein Erhöhen der Sicherheit bewirkt.
5. Da die *SINA*-Systeme aus Sicherheitsgründen nicht beschreibbar sind, müssen entsprechende Logging-Funktionalitäten zentral erfolgen. Dies ist mittels des Linux-Syslog-Daemons einfach zu realisieren. Entsprechende Meldungen laufen auf einem *Syslog / Revision-Server* im Management-System ein.

Weitere, in das Management-System integrierbare Komponenten sind ein *SNMP*[11]*-Gateway* sowie ein *NTP*[12]*-Server*. Somit wird ein zentrales Management aller Server im Netz ermöglicht, sowie die Option zur Verfügung gestellt, einen unabhängigen Zeitserver in das System zu integrieren.

[11] Simple Network Management Protocol, `RFC 1157`, u.a.
[12] Network Time Protocol, `RFC 958`, `RFC 1305`

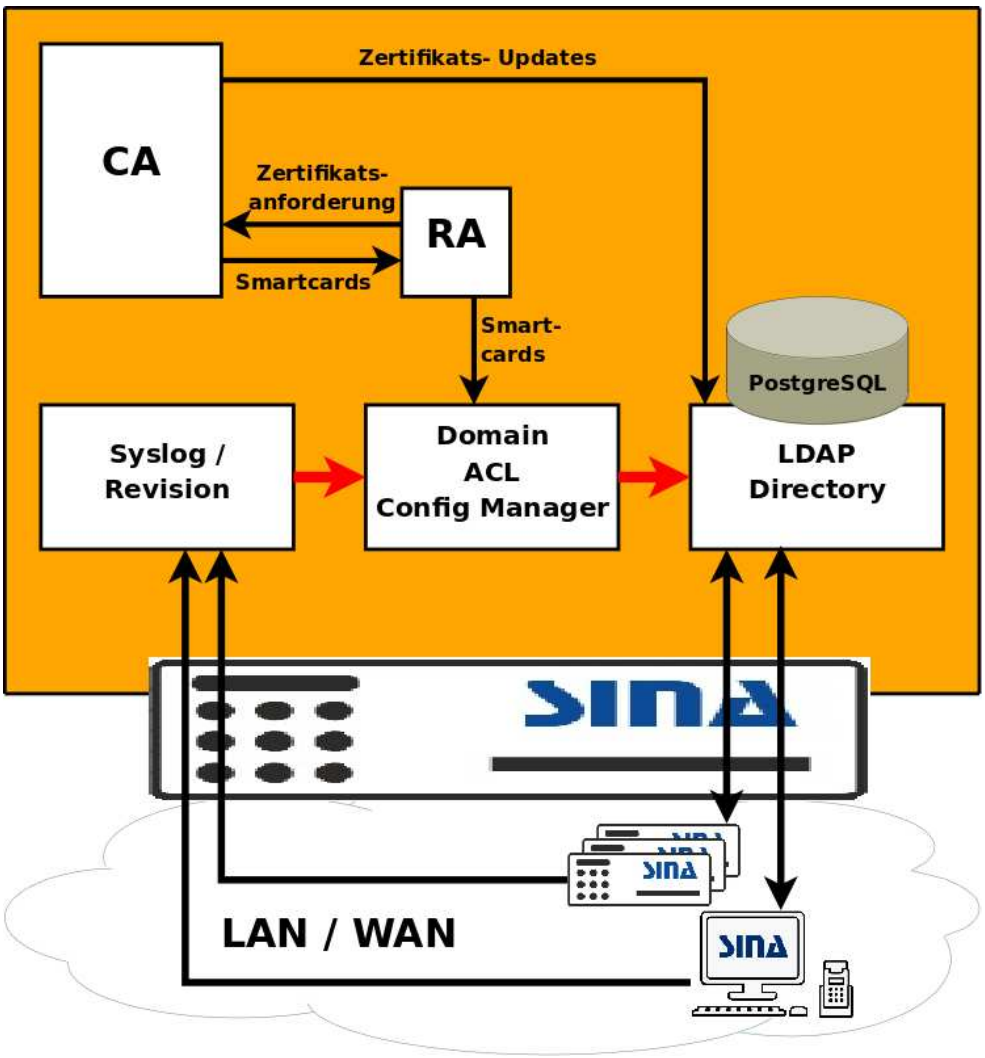


Abbildung 3.13: *SINA* Management

3.4 Zusammenfassung

Vernetzte Kommunikation ist durch Globalisierung, Kooperation und dem Bedarf an schnellen Reaktionszeiten nicht mehr wegzudenken. Hierbei müssen vielfältige Daten unterschiedlicher Schutzbereiche in Rechnern verarbeitet und gespeichert, und zwischen diesen ausgetauscht werden. Durch die aus ökonomischen Gründen entstehende Notwendigkeit, nicht-vertrauenswürdige Netze für den Transport der Daten zu verwenden, müssen diese gegen Zugriff und Manipulation geschützt werden. *IPsec* liefert die Grundlage für den sicheren Umgang mit Daten und regelt alle hierfür notwendigen Mechanismen. Hierauf basierend wurde die *SINA*-Produktpalette entwickelt, die gehärtete Systeme für verschiedene Schutzbereiche zur Verfügung stellt. Diese ermöglichen es, komplexe, mehrschichtige Sicherheitsszenarien zu implementieren, und bestehende Netze wirksam abzusichern.

3.5 Ausblick

Weitere Forschungsaspekte eröffnen sich insbesondere in folgenden Bereichen:

- **Schutz vor Angriffen von *Innen*:** Die Absicherung und Verschlüsselung der Kommunikation im *SINA*-Konzept dient maßgeblich dem Schutz der über nicht-vertrauenswürdige Netze geleiteten Kommunikation. Die eingestuften Netze werden in diesen Szenarien als sicher bzw. vertrauenswürdig in dem Sinne angenommen, dass keine weiteren internen Sicherungsmaßnahmen notwendig sind. Dies entspricht der Handhabung vieler eingestufter Firmen- und Behördennetze. Nicht Rechnung getragen wird hierbei die Gefahr durch Manipulationen oder Eingriffe von Innentätern. Abhängig der Untersuchungen und Quellen ist die Angabe des Anteils von Innentätern bezogen auf die Gesamtzahl von Sicherheitsvorfällen sehr unterschiedlich, in den meisten Fällen wird ein Wert von 50 Prozent angegeben. Während andere Studien von einem Anteil von bis zu 80 Prozent ausgehen, ermittelte eine Studie von *Verizon*[5] aus dem Jahre 2008 einen Anteil von lediglich 18 Prozent. Unabhängig des genauen Anteils zeigt sich jedoch die starke Gefährdung und somit der Handlungsbedarf, da insbesondere Innentätern oftmals einen einfachen Zugang zum internen Netz und entsprechende Kenntnis über die Infrastruktur und die eingesetzten Systeme haben.
- **Angriffsdetektion und -verteidigung in den Basis-Komponenten:** Neue Angriffsverfahren ermöglichen das Einbringen eines Hypervisors, welcher die Kontrolle des Systems übernimmt, ohne dass dies das eigentliche System registrieren kann. Beispiel hierfür ist das Projekt *Bluepill*[6]. Die *SINA*-Architektur muss auf Angriffsmöglichkeiten bezüglich dieser Verfahren hin geprüft werden, weiterhin gilt es zu untersuchen, ob derartige Konzepte in das *SINA*-System integriert werden können, um die Sicherheit und Verteidigungsfähigkeit der Architektur zu erhöhen.

Literaturverzeichnis

[1] Faltblatt SINA (Sichere Inter-Netzwerk-Architektur),
http://www.bsi.de/literat/faltbl/Sina.pdf

[2] SINA Systembeschreibung,
http://www.bsi.de/fachthem/sina/sysbesch/sysbesch.htm

[3] SINA Whitepaper Systemarchitektur,
SINA Daten und Fakten Sheet,
SINA Produktblätter,
www.secunet.de

[4] Security Engineering: A Guide to Building Dependable Distributed Systems,
Chapter 14, Physical Tamper Resistance, Figure 14.3,
http://www.cl.cam.ac.uk/ rja14/Papers/SE-14.pdf

[5] 2008 Data Breach Investigations Report,
Verizon Business,
http://www.verizonbusiness.com/resources/security/databreachreport.pdf

[6] Joanna Rutkowska,
http://invisiblethings.org/

Kapitel 4

Sicherheit durch Self-Protection und Self-Healing

Björn Stelte

Um die Komplexität des Managements verteilter und vernetzter Systeme beherrschen zu können, stoßen konventionelle Ansätze an ihre Grenzen. Als Alternative werden Ansätze aus dem Autonomic Computing wie z.B. Self-Protection oder Self-Healing als mögliche Lösungsansätze im Bereich IT-Security betrachtet. Obwohl erste Überlegungen bereits existieren, steht hier die Forschung erst am Anfang.
In dieser Ausarbeitung wird eine kurze Einführung in den Bereich des Autonomic Computing gegeben und Beispiele für Selbst-Heilungs-Prozesse im Bereich der Betriebssysteme dargelegt. Die Arbeit schließt ab mit einem kurzen Einblick über Selbst-Schutz und Selbst-Heilungsfähigkeiten von Protokollen für drahtlose Sensornetze (WSN).

Inhaltsverzeichnis

4.1 Einleitung

Durch den zunehmenden Grad an Komplexität von Computersystemen werden diese Systeme hinsichtlich manueller Kontrolle, Steuerung und Konfiguration immer schwieriger zu verwalten. Ein durch IBM initiierte Initiative greift genau an diese Problematik auf. Der Begriff "Autonomic Computing" wurde in diesem Zusammenhang von IBM durch das Autonomic Computing Manifesto im Jahre 2001 geprägt. Mittlerweile gewinnt Autonomic Computing immer mehr an Bedeutung, was durch die jährlich stattfindende IEEE Konferenz zum Thema und durch zahlreiche akademische Arbeiten belegbar ist. Zusätzlich ist zu erwähnen, dass mittlerweile neben IBM auch andere große IT Unternehmen wie HP und SUN der Initiative beigetreten sind und die Forschung im Bereich Autonomic Computing weiter vorantreiben.
Das Ziel der Autonomic Computing Initiative ist es Möglichkeiten aufzuzeigen mit denen auch die komplexesten Systeme beherrschbar bleiben. Neben einer Reduzierung des Aufwands für die administrativen Mitarbeiter soll unter anderem die Fehleranfälligkeit sowie Ausfallzeiten minimiert werden. Neben einer automatischen Konfiguration ist eine Anpassung von Parametern im laufenden Betrieb, Erkennen von Fehlern und Angriffen auf das System usw. als technisches Ziel formuliert worden. Ich werde in dieser Seminararbeit diese technischen Ziele hervorheben und Beispiele der Umsetzung in den Bereichen der Betriebssysteme und der Sensornetze zeigen.
Zunächst werde ich in Kapitel 4.2 den Begriff Autonomic Computing bzw Self-Management klären und deren Eigenschaften darstellen. Hierauf aufbauend werde ich mich in Kapitel 4.3 vor allem um die Aspekte Selbstheilung und Selbstschutz von Systemen beschäftigen. Da in diesem Seminar-band vor allem Sicherheitsaspekte im Vordergrund stehen, befasse ich mich in Kapitel 4.4 mit der theoretischen Bewertung von Sicherheit von IT-Systen um dann in Kapitel 4.5 Bespiele für den Einsatz von Selbstheilung unter dem Aspekt Sicherheit darzustellen. In Kapitel 4.5.1 werde ich Konzepte der Selbstheilung für Betriebssystem aufzeigen, während in Kapitel 4.5.2 kurz der Einsatz von Selbstheilungs- und Selbstschutzmethoden im Bereich der drahtlosen Sensornetze vorgestellt werden.
Beginnen möchte ich im nächsten Kapitel jedoch zunächst mit der Klärung des Begriffs Autonomic Computing.

4.2 Self-Management

4.2.1 Der Begriff Autonomic Computing

Durch die ständige zunehmende Komplexität der IT-Systeme werden diese immer schwieriger zu administrieren. Paul Horn, Director of Research bei IBM, beschrieb 2001 in seiner Arbeit "Autonomic computing: IBM's Perspective on the State of Information Technology" die Vision von Autonomic Computing als eine Lösung des Problems. Das Ziel, die Reduzierung des Management Aufwands, soll erreicht werden, durch eine Selbst-Verwaltung der Systeme. Der Benutzer kontrolliert das System nicht mehr direkt, sondern definiert Regeln, legt also im Vorhinein fest was bei einem Ereignis zu tun ist.

Als Vorbild diente IBM hierbei das menschliche Nervensystem. Lebenswichtige Funktionen wie die Atmung, Herzschlag oder Körpertemperatur werden ohne das Zutun vom Gehirn gesteuert. Genauso wie das Gehirn mit der Komplexität des menschlichen Körpers überfordert sein würde, so ist auch der System Administrator nicht in der Lage ein komplexes Computer System vollständig und in Echtzeit zu kontrollieren bzw managen. Ein Autonomic Computing System soll diese diese Problematik entschärfen, es stellt sich nun die Frage wie ein solches System aufgebaut ist und zunächst welche Eigenschaften es besitzen muss.

4.2.2 Was macht ein Autonomic Computing System aus?

self-aware Ein Autonomes System muss seine Komponenten, deren aktuellen Status, Kapazität sowie alle Verbindung zu anderen Systemen kennen. Diese umfassenden Kenntnisse über die verfügbaren Ressourcen sind nötig. Komponenten welche dem System nicht bekannt sind, sind auch nicht von ihm ansprechbar, nicht managebar.

self-reconfigurating Eine sich dynamisch verändernde Umgebung macht Anpassungen an der Systemkonfiguration nötig. Diese Anpassungen müssen automatisch stattfinden um eine optimale Konfiguration aus der komplexen Menge von möglichen Konfigurationen rechtzeitig zu finden. Ein System Administrator ist nicht in der Lage eine optimale Rekonfiguration innerhalb kürzester Zeit durchzuführen, bedenkt man, dass einige Minuten später wohlmöglich die Umgebung sich einmal mehr verändert hat und eine erneute Rekonfiguration des Systems ansteht.

self-optimizing Es muss ein ständiger Prozess der Selbst-Optimierung stattfinden, denn die Anforderungen an das System können sich laufend ändern. Um Veränderungen wahrzunehmen ist ein erweiterter feedback-control Mechanismus nötig, welcher dem System erlaubt Metriken zu beobachten und entsprechend zu handeln. Alle Komponenten des Autonomen Systems müssen dazu 'kontrollierbar' sein.

self-healing Probleme müssen erkannt werden und umgangen oder gelöst werden können. Ressourcen können auf eine andere Art und Weise genutzt werden oder das System ist anders zu konfigurieren. Redundante oder nicht ausgelastete Teile des Systems können als Ersatzelement dienen. Das System soll eine Beschädigung erkennen und selbstständig einen "Heilungsprozess" einleiten.

self-protecting Angriffe auf das System sollen selbstständig erkannt und abgewährt werden. Dem Benutzer bleiben die Analyse und Abwehr verborgen. Ein erkannter Systemangriff wird dem Administrator gemeldet und sofern möglich bereits automatisch Gegenmaßnahmen eingeleitet.

self-adapting Das Autonome System muß seine Umgebung kennen. Die Systeme müssen untereinander Kommunizieren, in der Lage sein sich selbst zu beschreiben und anderen Systemen ihre verfügbaren Ressourcen mitteilen können. Andere Systeme in der Umgebung müssen selbständig gefunden werden und deren Ressourcen bekannt gemacht werden können.

offen Unabhängigkeit eines Systems bedeutet auch, dass offene Standards verwendet werden müssen. Ein 'closed-source' System ist schwer zu adaptieren, kennt man doch die Implementierung nicht. Ein Autonomes System muß jederzeit in der Lage sein mit anderen art-unbekannten Systemen zu kommunizieren und Ressourcen auszutauschen.

autonom Die Komplexität soll vor dem Benutzer verborgen sein. Das System muß in der Lage sein Entscheidungen soweit wie möglich autonom zu treffen. Im Bedarfsfall muß das System den Benutzer selbständig informieren.

4.2.3 Eigenschaften des Autonomic Computing

Das Forschungsgebiet Autonomic Computing hat zum Ziel Methoden aufzuzeigen, welche es ermöglichen dass Computer Systeme weitgehend autonom handeln. Das autonome Handeln ermöglicht es den Verwaltungsaufwand komplexer Computer Systeme zu reduzieren. Die geforderten zentralen Eigenschaften eines autonomen Computer Systems sind hierbei die Folgenden.

self-healing Reduktion von Ausfällen des Systems mittels automatischer Fehlererkennung und Reaktion auf Fehler.

self-protection Schutz des Systems vor unerwünschten Veränderungen.

self-configurtion Der Aufwand an manueller Konfiguration soll reduziert werden, Parameter werden automatisch angepasst.

self-optimization Angemessene Reaktion auf Umgebungsänderungen um das Leistungsziel zu erreichen.

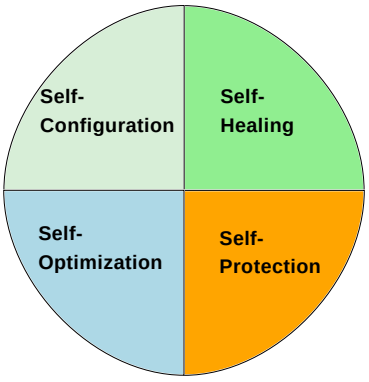


Abbildung 4.1: Eigenschaften des Autonomic Computing nach IBM.

Insbesondere sind die Aspekte Selbstheilung und Selbstschutz für Computer Systeme und insbesondere auf Ebene der Betriebssysteme im Hinblick auf Zuverlässigkeit und Sicherheit von Bedeutung. Durch Änderungen der Umgebungen oder gar Angriffe auf das System sind darauf autonom handelnde Systeme einem normalen System im Vorteil. Je nach

Grad der Automatisierung ergibt sich somit nicht nur eine Eingrenzung der Komplexität bezüglich des Konfigurationsaufwandes, sondern es ermöglicht auch neue Möglichkeiten im Bereich der Sicherheit von Computer Systemen.
Es stellt sich jedoch die Frage, wie und ob diese Eigenschaften zu implementieren sind bzw. bis zu welchem Grad eine Automatisierung möglich ist.

4.2.4 Automatisierungsgrad

Der Grad der Automatisierung gibt an, wie autonom das System arbeitet bzw. inwieweit eine Interaktion mit dem Benutzer stattfindet. Der Automatisierungsgrad adaptiert nach Sheridan gibt eine grobe Skala an:

1. Computer offers no assistance, human must do it all.
2. Computer offers a complete set of action alternatives, and
3. narrows the selection down to a few, or
4. suggests one, and
5. executes that suggestion if the human approves, or
6. allows the human a restricted time to veto before automatic execution, or
7. executes automatically, then necessarily informs the human, or
8. informs him after execution only if he asks, or
9. informs him after execution if it, the computer, decides to.
10. Computer decides everything and acts autonomously, ignoring the human.

Der niedrigste Automatisierungsgrad entspricht einem System, welches von dem Benutzer gänzlich konfiguriert und überwacht werden muss. Desto höher der Automatisierungsgrad, desto weniger Eingreifen eines Benutzers wird benötigt um ein System lauffähig zu halten (Sollzustand). In einem niedrigen Automatisierungsgrad kann ein System dem Nutzer Handlungsvorschläge machen oder zumindest bei Fehlverhalten hierüber informieren. In den Varianten des höheren Automatisierungsgrades wird das System selbstständig agieren und eine Aktion des Benutzers ist nicht vorgesehen bzw. nicht ohne weiteres möglich. Die Systemvariante mit niedrigem Automatisierungsgrad lässt dem Benutzer eine höhere Kontrolle über den Handlungsablauf als die Variante mit hohem Automatisierungsgrad, die kaum Kontrollmöglichkeiten bietet.

4.3 Aspekte der Selbstheilung / des Selbstschutzes

Der Begriff Selbstheilung wird häufig gleichgesetzt mit Fehler-Toleranz. Koopman zeigt in [2], dass eine Gleichsetzung nicht zweckmäßig ist. Die wichtigsten Aspekte der Selbstheilung lassen sich ferner in vier Kategorien einteilen, Fehlermodell, Systemvollständigkeit, Systemreaktion und Zusammenhänge im Design.

4.3.1 Fehlermodell

Um festzustellen ob ein System fehlertolerant ist und im welchen Umfang, wird ein Fehler Modell oder Fehler Hypothese erstellt. Es wird anhand einer Spezifikation festgelegt gegenüber welchen Fehler-Arten das System tolerant ist. Ein sich selbstheilendes System benötigt ein Fehlermodell, ohne kann nicht bestimmt werden ob eine Selbstheilung in bestimmten Situation erfolgen kann oder nicht.
Relevante Eigenschaften des Fehlermodells sind fault duration, fault manifestation, fault source und granularity.

Fault Duration Zeitlich gesehen können Fehler permanent, in unregelmäßigen Zeitabständen oder auch nur spontan auftreten. Um eine Fehler Behandlung zu ermöglichen, ist es erforderlich die Maßnahmen einer Selbstheilung an die Annahmen der zu erwarteten Dauer eines Fehlers anzupassen.

Fault Manifestation Auftretende Fehler haben unterschiedliche Symptome, die es zu unterscheiden gilt. Einige Fehler sind harmloser und andere schwerwiegender, der Einfluss auf das System ist also zu betrachten. Es kann nützlich sein das Erkennen eines Fehlers anderen Systemen mitzuteilen, wobei es in anderen Situationen wohlmöglich unnötig ist.

Fault Source Abhängig von der Quelle des Fehlers stehen verschiedene Selbstheilungsalgorithmen zur Auswahl. Fehler können in der Implementierung, Spezifikation, Bedienung des Systems usw liegen. Andersherum können einige Selbstheilungsalgorithmen nur bestimmte Fehler begegnen, z.B. nur Hardwarefehler wie den Verlust von Speicher oder CPU Kapazität und nicht Softwarefehler.

Granularity Die Auswirkung des Fehlers auf das System ist entscheidend für die Wahl des richtige Selbstheilungsalgorithmuses. Ein einzelner Fehler mag ein Softwaremodul, einen Task, ..., oder ein gesamtes System betreffen. Je nach schwere des Fehlers sind entsprechende Gegenmaßnahmen zu treffen.

Fault Profile Expectations Mithilfe eines Fehler-Profils lassen sich die Fehler einordnen. Neben der Annahme der Fehlerquelle ist auch ihre Auftrittswahrscheinlichkeit von Bedeutung. Eine mögliche Klassifizierung ist die Einteilung in erwartete, vermutete und unerwartete Fehler, welche wieder als unter Kategorie eingeteilt werden können in zufällige, abhängige und beabsichtigte Fehler.

4.3.2 Systemreaktion

Bevor ein System auf Fehler reagieren kann, muss zunächst einmal der Fehler detektiert und klassifiziert werden. Die Reaktion des Systems wird beeinflußt von den Rahmenbedingungen des Systems, der Anwendungsdomaine, sowie hinsichtlich Zuverlässigkeit und Sicherheit.

Fault Detection Ohne zuverlässige Erkennung eines Fehlers kann ein System nicht auf diesen reagieren. Verschiedene Ansätze der Fehlererkennung existieren, hierzu zählen:

- Erkennung innerhalb einer Komponente
- Stetiger Vergleich redundanter Komponenten
- Peer-to-Peer Überprüfung
- Überprüfung durch Supervisor (Leitender)

Neben diesen Ansätzen sind auch die Vorgehensweisen von Interesse, hierzu zählen:

- passive Überprüfung der Ergebnisse
- Audit Tests oder check tasks
- redundante Ausführung der Berechnung
- on-line Selbsttest
- periodischer Reboot

Es sollte sichergestellt sein, dass Fehler innerhalb einer gesetzten Zeitspanne erkannt werden. (hard-realtime Bedingung)

Degradation Selbst nach erkennen eines Fehlers ist es nicht sichergestellt, dass ein System sofort vollständig wiederhergestellt werden kann. Für einige Anwendung mag ein teilweise wiederhergestelltes System reichen, für andere nicht. Funktionalitäten können wohlmöglich auf andere Systeme ausgelagert werden oder aber das System kann sich auf wenige Kernfunkionalitäten beschränken damit kritische Dienste überleben können.

Fault Response Nach der Detektierung des Fehlers muss das System angemässen reagieren, folgende Reaktionen sind möglich:

- Fehlermaskierung
- Rollback zu einem Checkpoint
- Rollforward mit Ausgleich des Fehlers
- Wiederholung der Operations Ausführung mit original und alternativen Ressourcen
- Rekonfiguration der Systemarchitektur on-the-fly oder per reboot
- Ausführung alternativer Tasks

- Beenden von niedrig priorisierten Tasks
- Anforderung von externer Hilfe (Ressourcen)

Die Reaktion des Systems sollte optimiert sein bezüglich Korrektheit, Quality-of-Service, Sicherheit und Integrität. Allgemein lässt sich die Reaktion unterscheiden in die Kategorien präventiv, pro-aktiv und reaktiv.

Recovery Nach Detektion und Reaktion auf den Fehler muss das System wieder in den normalen Betriebszustand zurückkommen. Gegebenenfalls müssen neu hinzugekommene Ressourcen eingegliedert oder initialisiert werden, neu-gestartete Prozesse ihren alten Zustand mitgeteilt werden, usw. Beispielsweise kann es vorkommen, dass eine Komponente eine cold reboot benötigt um den recovery Prozess abzuschließen, einer anderen recht vielleicht ein warm reboot aus.

Time constants Unter der Beachtung des zeitlichen Aspekts sind einige Selbstheilungsalgorithmen attraktiver wie andere. So kann ein Reboot Prozess zu einem recht langen Ausfall der Dienste führen. Andere Maßnahmen wie das Beenden und erneute Aufrufen der fehlerhaften Prozesse dagegen kann sehr viel schneller zu einem Erfolg führen, unter der Annahme, dass die Maßnahme ausreicht. Aspekte wie die Frequenz der Sicherung des Systemstatus oder die Dauer eines System-Reboots sind entscheidend bei der Frage nach der richtigen Wahl des Selbstheilungsalgorithmuses.

Assurance Randbedingungen eines Systems müssen jederzeit, also während der Recovery Phase und nach erfolgreicher Selbstheilung, sichergestellt sein.

4.3.3 Systemvollständigkeit

Der Umgang mit dem eingeschränkten Wissen, unvollständigen Design sowie unvollständiger Spezifikation eines Systems ist von Bedeutung.

Architectural completeness Architekturen von Systemen sind einem ständigen Wandel unterzogen. Neue Komponenten kommen hinzu oder werden verändert. Die Implementierung dieser Komponenten werden 'gepatched', neu-entwickelt, was zu Problemen hinsichtlich einer veränderten Konfiguration führen kann usw.

Designer knowledge Gerade komplexere Systeme werden von einem Team von Designer und Programmieren entwickelt. Jeder von ihnen arbeitet in seinen eigenem Teilbereich und kennt das gesamte System nur im groben. Oftmals werden die Komponenten aufgrund von Abstraktionen und unvollständiger Dokumentation des Gesamtsystems nur hinsichtlich der Testcases des Pflichtenhefts hinentwickelt.

System self-knowledge Selbstheilende Systeme müssen mit der Tatsache, dass sie sich über die Zeit verändern, die Konfiguration sich den Gegebenheiten anpasst. So kann es z.B. sein, dass ein System einen Neustart in nächster Zeit benötigt und dieses den anderen Systemen mitteilt, damit sich diese darauf einstellen können und z.B. nicht zur selben Zeit selber einen Neustart einplanen. Das System muß dazu die Veränderungen bei der Auswahl der Selbstheilungmethode mit einbeziehen. Die Veränderungen entstehen aus Anpassungen an die Umwelt, dem Verwendungszweck des

Systems, Adaptierung von externen Ressourcen, Veränderung im Nutzerverhalten oder vorhersehbare Ressourcen Engpässe und Ausfälle.

4.3.4 Zusammenhänge im Design

Es gibt einige Faktoren, welche die Möglichkeiten des Selbstheilungsprozess beeinflussen.

Abstraction level Ein selbstheilendes System kann auf verschiedenen Abstraktionsebenen Selbstheilungsmethoden anwenden. So gibt bezüglich einer Applikation die Möglichkeiten eines Middleware Mechanismus, auf der Betriebssystem Ebene oder Hardware Ebene. Daneben können Selbstheilungtechniken in der Implementierung verwendet werden (z.B. Wrapper) oder in der Architektur des Systems.

Component homogeneity Systeme und auch Komponenten der Systeme sind oft nicht homogen sondern heterogen. Verschiedene Versionen der verwendeten Software aber auch der Hardware, verschiedene Betriebssysteme, usw. müssen bei der Auswahl der Selbstheilungmethode mit bedacht werden. Die Heterogenität kann den Grad der möglichen Heilung einschränken, z.B. bei der Migration von Prozessen von einem System auf ein anderes heterogenes System muß das z.B. das Konfigurations Management vor und nach der Migration bedacht werden, bzw. eine Migration überhaupt möglich sein.

Behavioral predetermination Die meisten Systeme haben kein perfektes deterministisches oder nicht-deterministisches Verhalten. Algorithmen zur Selbstheilung müssen diese Verhalten berücksichtigen und entsprechend reagieren. Die Algorithmen können dabei selber wieder nicht-deterministische Verhaltensmuster aufweisen.

User involvement in healing Systeme sind, auch wenn wir es uns noch so wünschen, limitiert in ihren Möglichkeiten vollkommen autonom sich selbst zu heilen. Die Interaktion mit dem Benutzer ist immer dann erforderlich, wenn das System keine eindeutige Möglichkeit oder überhaupt keine Möglichkeit der Selbstheilung hat. Der Benutzer kann dabei dem System Hilfestellung geben oder die entsprechenden Entscheidungen für das System treffen.

System linearity Komponenten sind oft untereinander abhängig. Diese Abhängigkeit halt Folgen für die gewählte Selbstheilungsmethode. So kann die falsche Wahl eine Kette von neuen Problemen oder Ausfällen hervorrufen. Bei einem linear aufgebauten System kann eine Selbstheilung ohne Rücksicht auf eventuelle Auswirkungen auf andere Komponenten erfolgen. Inwieweit ein System linear aufgebaut ist hängt von seiner Architektur ab. Je nach Grad der Kommunikation zwischen den Komponenten ist eine stärker oder schwächere Auswirkung auf den Selbstheilungsprozess festzustellen. Bei der Selbstheilung auftretende Effekt und Einflüsse auf andere Komponenten, z.B. bei der Interaktion, müssen berücksichtigt werden.

System scope Je nach Größe des zu betrachtenden Systems sind nicht alle denkbaren Selbstheilungmechanismen möglich. Ein System bestehend aus einer Komponente hat nicht die selben Möglichkeiten wie Systeme mit vielen Komponenten. Kategorien können hierbei unter anderem sein:

- einzelne Komponente
- Rechnersystem
- Rechnersystem mit Benutzer
- Enterprise Automation Suit

4.4 Gemeinsame Kriterien für die Bewertung der Sicherheit von Informationstechnologie

Die Common Criteria (CC) nach Norm ISO/IEC 15408 ist ein Katalog für gemeinsame Kriterien für die Prüfung und Bewertung der Sicherheit von Informationstechniken. Es ist somit ein Standard über die Kriterien der Bewertung und Zertifizierung der Sicherheit von Computersystemen.
Für den Einsatz in bestimmten Umgebungen wird die Zertifizierung nach Common Criteria oft verlangt, hierzu zählen meist militärische Applikation im Bereich der Eingebetteten Systeme wie z.B. Steuerungseinheiten für die head-up Anzeige (HUD) eines Kampfjets [z.B. Green Hills RTOS System]. Die in der Common Criteria definierten sieben Stufen Evaluation Assurance Level 1 (EAL1) bis EAL 7 beschreiben die Korrektheit der Implementierung, kurz ihre Vertrauenswürdigkeit. Jede dieser Funktionalitätsklassen beschreibt somit eine bestimmte Grundfunktion der Sicherheitsarchitektur. Die Korrektheit der Implementierung des betrachteten Systems auf Basis der Einordnung in einer der sieben Stufen sagt aus, wie ein Programm gegenüber einer Spezifikation validiert wurde. Hier liegt jedoch auch ein Problem der Common Criteria, da nicht spezifizierte Fähigkeiten auch nicht geprüft werden (z.B. Echtzeitfäigkeit eines Systems).
Die EAL Stufen sind ein Maß für den benötigten Aufwand zur Überprüfung ob ein System oder Programm die geforderten Sicherheitsanforderungen erfüllt. Hierbei ist anzumerken, dass um so komplexer ein System ist, desto schwieriger ist es dieses System nach einer der höheren Stufen zu zertifizieren. Die Komplexität der Systeme muß überschaubar bleiben um nach Common Criteria eine erstrebenswerte Zuverlässigkeit bescheinigt zu bekommen. Eine Reduktion der Komplexität der Systeme durch den Einsatz von Autonomic Computing Methoden gewinnt hier an Bedeutung.

Die EAL Stufen nach Common Criteria sind:

- EAL1 Functionally Tested
- EAL2 Structurally Tested
- EAL3 Methodically Tested and Checked
- EAL4 Methodically Designed, Tested, Reviewed
- EAL5 Semiformally Designed and Tested
- EAL6 Semiformally Verified Design and Tested
- EAL7 Formally Verified Design and Tested

Für die Zertifizierung von Betriebssystemen nach der Common Criteria werden Audit-Subsysteme benötigt. Diese bieten die Möglichkeit viele Ereignisse eines Betriebssystems genau zu protokollieren. Als Beispiel im Falle von Linux seien hier nur die Erweiterungen SELinux und AppArmor stellvertretend genannt.
Ein autonomes Betriebssystem benötigt die Informationen eines Audit-Subsystems um nach einer Diagnose entsprechend den Self-CHOP Regeln zu handeln. Im nächsten Kapitel möchte ich näher auf Möglichkeiten von Selbstheilung und Selbstschutz von Betriebssystemen eingehen.

4.5 Neue Lösungsansätze für die IT-Security

4.5.1 Betriebssystem Ebene

Stellt man die Frage, wie zuverlässig ein Computersystem ist, so muss man diese Frage in Wandel der Zeit sehen. Computer-Benutzer vor 20 Jahren waren meist Computerexperten, die bei einem Fehler genau wussten was sie zu tun haben. Die Erwartungshaltung jener Tage war eine andere wie heute. Ein durchschnittlicher Benutzer unserer Tage vergleicht einen Computer eher mit Geräten wie Radio, Fernseher oder dergleichen. Er erwartet, dass das Geräte (hier der Computer) jahrelang ohne Ausfälle funktioniert, genauso wie er es beispielsweise von seinem Fernseher gewohnt ist. Leider ist dieses nicht immer der Fall, was jedem Computer Benutzer heutzutage leidlich wohl-bekannt ist.
Der Grund hierfür liegt hierbei oftmals in der Komplexität heutiger Rechner und insbesondere der Software. Die Software wie das nötige Betriebssystem enthält Fehler, und je mehr und je Komplexer die eingesetzte Software ist, desto mehr Fehler gibt es.
Da in diesem Seminar vorallem die Sicherheit von Systemen im Vordergrund steht werde ich mich in diesem Kapitel auf den Bereich der Betriebssystem und Möglichkeiten deren Absicherung konzentrieren.

Armored Operating System

Der größte Teil eines Betriebssystems machen Gerätetreiber und Dienste aus. Doch genau diese Gerätetreiber sind häufig fehlerträchtig. Dies liegt zum einen an der Komplexität von Gerätetreibern und zum anderen wie Tanenbaum ausführt nicht so häufig einem Reviewprozess unterzogen werden wie zum Beispiel ein Scheduler.
Da Gerätetreiber ein Hauptgrund für Ausfälle von Computersystemen sind, ist das Ziel genau diese Ausfälle zu vermeiden und damit die Zuverlässigkeit des Systems erhöhen.
Swift et al. stellt dazu in seiner Arbeit [1] das Konzept eines Betriebssystems vor, welches mit Ausfällen von Gerätetreibern zurechtkommt indem Treiber bei Fehlverhalten neugestartet werden.
Aufgebaut ist Swifts Arbeit auf dem 'Nooks' Konzept. Nooks sind Subsysteme zur Kapselung von Gerätetreibern im Betriebssystemkern. Treiber haben eine so genannte Protection Domain, die Kommunikation zwischen Kernel und Treiber wird verfolgt, sowie Objekte des Treibers werden im Kernel verfolgt. Die von Swift vorgestellte Nooks Erweiterung für

Linux kommt mit Ausfällen von Treibern zurecht.
Das Problem von Nooks ist jedoch, dass nur der Kernel geschützt wird, Applikationen

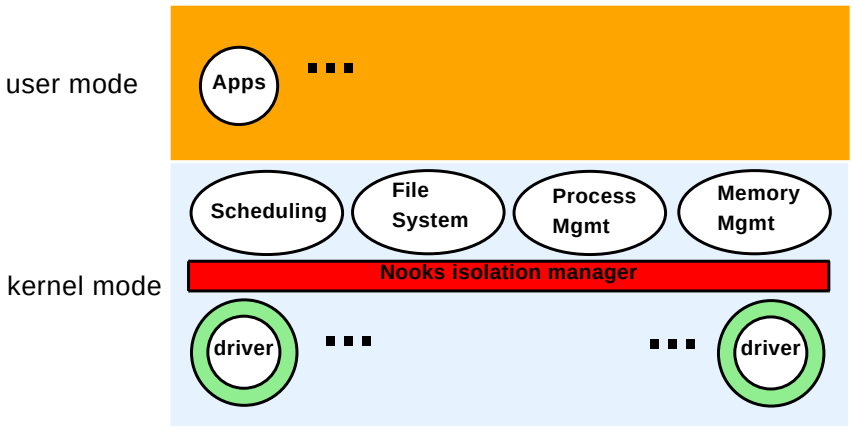


Abbildung 4.2: Nooks Architektur nach Swift.

verhalten sich weiterhin fehlerhaft. Hieraus ergab sich für Swift et al. daher die Erweiterung zum Konzept von Schadow Drivers.
Neben der aus Nooks bekannten Verfolgen den Zustand des Treibers, kann bei erkennen eines Fehlverhaltens der Treiber neu gestartet werden und im Gegensatz zu Nooks der Zustands des Treibers nach dessen Neustart wiederhergestellt werden. Somit besitzt das Shadow Driver Konzept neben der passiven auch eine aktive Komponente.
Neue Shadow Driver Instanzen werden durch eine Schadow Manager Prozess erzeugt und verwaltet. Von Nooks gemeldete Fehler der Treiber werden von Shadow Manager entgegengenommen und ausgewertet. Bei einer positiven Fehlermeldung setzt der Shadow Manager den entsprechenden Shadow Driver in den aktiven Betriebsmodus und leitet anschließend die Wiederherstellung des Treibers im Shadow Driver ein. Nach erfolgreichen Vorgang wird der passive Betriebsmodus für den Shadow Driver eingestellt.
Im passiven Betrieb protokolliert der Shadow Driver Anfragen an den Treiber. Hierzu merkt er sich den Zustand der Verbindungen, laufende Anfragen, Konfigurationsanfragen und verfolgt alle angeforderten Ressourcen wie Interrupts usw. Der passive Betrieb ist ein reines Monitoring, der Treiber wird direkt aufgerufen nur seine Kommunikation (Parameter und Ergebnisse) wird an den Shadow Driver weitergeleitet.
Im aktiven Betrieb kommunizieren Kernel und Treiber nicht direkt miteinander sondern über den Shadow Driver. Ein von Nooks bemerkter Ausfall wird an den Shadow Manager weitergegeben welcher den Shadow Driver in den aktiven Modus schaltet. Der Shadow Driver hält zunächst den betroffenen Treiber an, initialisiert diesen neu und stellt anschließend den Zustand des Treibers vor dem Ausfall wieder her.
Der Ablauf des aktiven Modus werde ich hier nun detailliert schildern. Zunächst wird die Ausführung des Treibers beendet und das entsprechende Gerät deaktiviert indem Interrupts und I/O Speicherbereiche freigegeben werden. Der Shadow Driver übernimmt die Stelle des Treibers, wird zwischen eigentlichen Treiber und dem Kernel dazwischen geschaltet. Objekte im Kernel, welche nötig sind um den Treiber anzusprechen, werden gehalten und alle übrigen Objekte werden freigegeben. Dieses ist eine Garbage Collection der Ressourcen des Treibers. Hiernach wird der Initialzustand des Treibers hergestellt. Dies geschieht indem der Shadow Driver eine Kopie des Datenbereichs, welches er

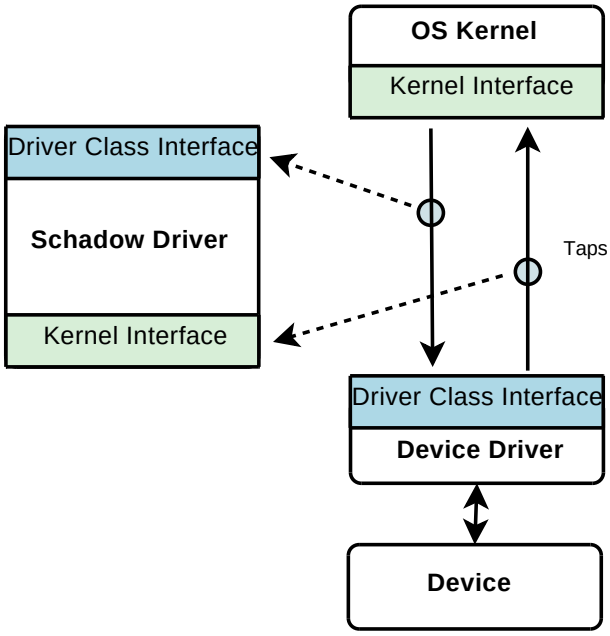


Abbildung 4.3: Shadow Driver im passivem Modus. Die Kommunikation zwischen Gerät und Kernel wird über die Taps dem Shadow Driver übermittelt.

beim Einfügen eines Treibers gemacht hat, verwendet. Da der Programmcode nicht verändert wurde, kann dieser wiederverwendet werden. Die Initialisierung des Kernels für den entsprechenden Treiber wird wiederholt und wie zuvor mitprotokolliert. Vorhandene Ressourcen aus dem Kernel werden wieder zugeteilt und Hardware Ressourcen wieder aktiviert. Der Shadow Driver verhält sich nun gegenüber dem Kernel so wie zum Zeitpunkt des ersten Ladens des Treibers. Um den Zustand des Treibers kurz vor dem Ausfall wiederherzustellen, werden entsprechende protokollierte Anfragen wiederholt. Hierzu müssen Verbindungen wieder geöffnet und konfiguriert werden sowie noch nicht bediente Anfragen an den Treiber geleitet werden. Der Zustand des Treibers vor dem Ausfall wird wiederhergestellt. Das Problem hierbei ist, dass bedingt durch den Ausfall der Zustand wohlmöglich nicht komplett wiederherstellbar ist. Der Treiber könnte abgestürzt sein nachdem er eine Anfrage an das Gerät gesendet hat jedoch bevor dir Aktion als abgeschlossen vermerkt wurde. Hierbei würde eine Wiederholung des Vorganges zu einem Duplikationsfehler führen, jedoch andersherum ein Verwerfen möglicherweise zu einem Datenverlust führen. Es ist daher zu unterscheiden welches Gerät angesprochen wurde, in wie in einem solchen Fall zu entscheiden ist. Wenn der Zustand erfolgreich wiederhergestellt wurde, so wird der Treiber wieder im Kernel eingebunden und ist bereit für neue Anfragen. Der Shadow Driver meldet dem Shadow Manager das Ereignis, welcher dem Shadow Driver danach wieder in den Monitoring also passiven Modus versetzt.
Das Problem beim Shadow Driver Konzept ist, dass nur Treiber Fehler der Art 'Fail Stop' behandelt werden. Diese Einschränkung führt dazu, dass nicht alle Fehler erkannt werden. Dies kann passieren, da beispielsweise ...

- Unzulässige Parameter wurden nicht als solche erkannt
- Fehler in der Kommunikation Treiber/Gerät

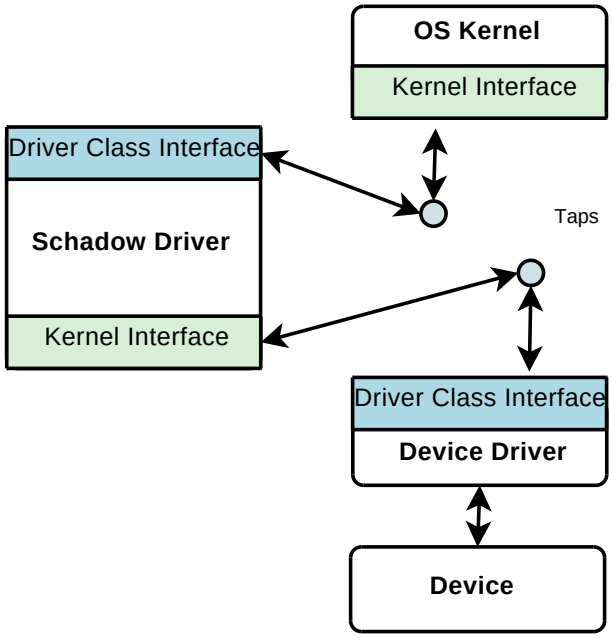


Abbildung 4.4: Shadow Driver im aktivem Modus. Die gesamte Kommunikation zwischen Gerät und Kernel wird von den Taps zu dem Shadow Driver umgeleitet.

- System hat nicht beendete E/A Anfragen nicht erkannt

Zur Verbesserung des Shadow Driver Konzeptes müsste die Semantik der Treiber stärker in die Fehlererkennung einbezogen werden. Dieses ist jedoch schwierig da die Komplexität der Treiber hoch ist.
Ferner sind deterministische Fehler, also Fehler welche immer wieder auftreten z.B. nach der Neuinitialisierung des Treibers, durch den Shadow Driver nicht abfangbar. Weiter Einschränkungen sind, dass das deterministische Laden und Entladen des Treibers muss möglich sein muss. Nur explizite Kommunikation kann verfolgt werden. Die Fehlfunktion des Treibers darf keine Seiteneffekte, wie das schreiben von falschen Daten auf die Festplatte, haben.

Microkernel

Nach Tanenbaum sind Microkernel eine, nach seiner Auffassung sogar die beste Lösung, um Betriebssysteme zuverlässiger zu gestalten. Angefürte Beispiele für Betriebssystem mit Microkernel sind MinixV3, L4 bzw. der Neutrino-Microkernel. Bei diesen Systemen wird Code aus dem Kernelbereich in den Userspace verlagert. Vor allem Gerätetreiber, welche häufig zu Systemabstürzen führen, werden hierdurch nicht im Kernelbereich ausgeführt. Die Komplexität des Kernelbereichs nimmt ab und somit ist nach Tanenbaum eine Verifikation des Codes durch einen Reviewprozess möglich. Ein schadhafter Gerätetreiber kann so keinen größeren Schaden anrichten. Dieses Designprinzip verfolgt das Minix Betriebssystem.
Anders als bei bekannten Betriebssystemen wie Linux oder Microsoft Windows, wird

in Minix Gerätetreiber und Dienste als normale Userspace-Prozesse mit eingeschränkten Rechten ausgeführt.

Monolitische Kernel wurden lange wegen ihrer besseren Performance gegenüber Micro-

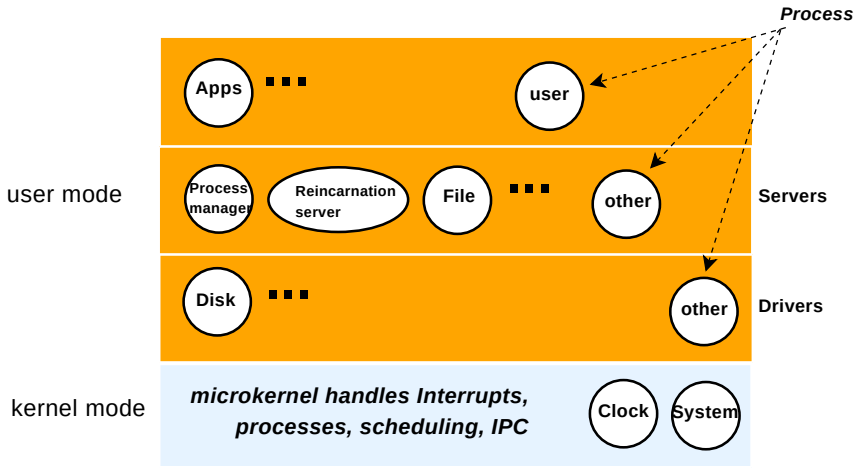


Abbildung 4.5: Minix Architektur nach Tanenbaum.

kerneln bevorzugt, jedoch zeigen neuere Entwicklungen, dass Zuverlässigkeit wichtiger wird wie Performanceunterschiede. Microkernel gelten als zuverlässiger wie Monolitische Kernel. Der Monolitische Kernel integriert alle Funktionen, die zur Systemsteuerung möglicherweise benötigt werden. Der Microkernel koordiniert lediglich die Zusammenarbeit der modular ausgelegten Subsysteme und zeichnet sich aus mittels Isolation durch Adressräume, geringe Komplexität, hohe Modularität. Beim Minix Microkernel liegt über dem Kernel der 'device driver layer', darüber der 'server layer'. Der 'reincarnation server' ist Eltern-Prozess aller anderen Server- und Treiberprozesse. Wird durch das ausbleiben der Antwort auf periodische pings ein Versagen eines Prozesses festgestellt, so führt dies zu einer Beendigung und Neustart des 'servers' (bzw des entsprechenden Prozesses). Zuverlässigkeit wird in Minix durch die geringe line-of-code (LOC) erreicht, alle Gerätetreiber sind user Prozesse. Tanenbaum führt weiter an, dass kein Fremdcode im Kernelbereich läuft und dass eine einfache Verifikation des Kernels daher möglich ist. Design-technisch wird buffering nicht benötigt, es ist kein buffer Management im Kernel vorhanden. Wie erwähnt werden die meisten Betriebssystem Prozesse in den user space verschoben. Bei einer Fehlfunktion ist daher nicht der Kernel betroffen, Tanenbaum nennt dieses 'user mode restriction'. Die user-mode Treiber und Serverprozess laufen nicht mit superuser Rechten und haben einen eigenen Adressraum, mit Ausnahme der eigenen Kernelaufrufe. Schadhafter Code wird nicht sofort ausführbar, da der Kernel Programme nur ausführt wenn der Kernel im 'instruction space' ist. Der 'reincarnation server' wird für die Selbstheilung benötigt. Nicht nur Abstürze durch schlechte Implementierung werden ausgeglichen, sondern auch erfolgreiche Angriffe wie DOS Attacken abgewährt oder zumindest erschwert.

Virtualisierung

Die Virtualisierung zeigt Methoden auf wie sich Ressourcen (CPU, Speicher, usw.) eines Rechners aufteilen lassen. Gerade im Bereich der Betriebssystem-Virtualisierung (bzw.

System Virtualisierung) hat in den letzten Jahren ein wahrer Boom eingesetzt. Ziel dieser Virtualisierung am Markt ist es oft die Auslastung eines Rechners vor allem in Rechenzentren zu erhöhen in dem mehrere virtuallisierte Rechner (Gastsysteme) in Form von virtuellen Maschinen (VM) auf einem physikalischen Rechner laufen. Dies erhöht die Wirtschaftlichkeit bedenkt man, dass zum Beispiel ein typischer Webserver die meiste Zeit nur einen Bruchteil seiner ihm zugewiesenen Ressourcen benötigt.
Hier werde ich mich jedoch mit einem anderen Aspekt der Virtualisierung befassen. Durch die Trennung von Ressourcen und deren Zuordnung zu Teilen des Systems (hier Virtuellen Maschinen) werden auch hinsichtlich Sicherheit und Zuverlässigkeit neue Möglichkeiten aufgezeigt. Beispiele für Systeme bzw Prototypen welche Virtualisierungstechniken zur Erhöhung der Sicherheit verwenden sind zu nennen,

- L4Linux bzw kqemu-L4
- TUD:OS
- Fiasco-Mikrokern2

Allen Varianten gemein ist, dass die jeweiligen virtuellen Systeme isoliert voneinander ausgeführt werden. Eine fehlerhafte Anwendung oder gar Kernel eines Gastsystems hat keinen Einfluss auf anderes System des gleichen Host-Rechners, da die Ressourcen reserviert sind. Da für laufende Prozesse des Gastsystems die Virtualisierung transparent ist, kann ein Schadprogramm nicht erkennen ohne weiteres Erkennen, dass es innerhalb einer virtuellen Maschine ausgeführt wird. Der darunter liegende VM Monitor (Hypervisor bei einer Paravirtualisierung) jedoch könnte die Auswirkungen des Schadprogramms erkennen und entsprechend gegensteuern. Das Schadprogramm kann in jeden Fall erst einmal nur die virtuelle Maschine verändern, der Schaden ist somit bezüglich des Gesamtsystems begrenzt. Weder Fehlverhalten noch erhöhter Ressourcenbedarf kann die Funktion eines anderen Gastsystems beeinträchtigen. Dabei ist das Betriebssystem als Gastsystem auf einer virtuellen Abstraktionsschicht vergleichbar mit dem Prozessmodell eines Betriebssystems mit prämtivem Multitasking.
Mögliche Virtualisierungmethoden sind Emulation und native execution mit und ohne Hardware Unterstützung (Native Virtualisierung und Paravirtualisierung).

Bei der *Emulation* gibt es die Möglichkeit der Interpreterschleife (wie z.B. bei Bochs) oder die dynamische Übersetzung (z.B. QEMU). Die komplette Hardware wird abstrahiert und emuliert. Die Emulation wird häufig nur dann eingesetzt wenn das Gastsystem eine andere Hardwarearchitektur wie das Hostsystem verlangt, da die Emulation einiges an Ressourcen bindet und somit langsam ist.

Bei der *native execution* unterscheidet man Host-Mechanismen und Host-Kernerweiterungen (z.B. VMWare, kQEMU).
Bei der nativen Virtualisierung wird nur ein Teil der Hardware emuliert, genau soviel, dass ein unmodifiziertes Betriebssystem als Gastsystem ausgeführt werden kann. Der andere Teil der Hardware wird über eine Verwaltungsschicht direkt angesprochen.

Bei der *Paravirtualisierung* erfolgt Keinerlei Emulation von Hardware, alle Befehle werden direkt über die Verwaltungschicht ausgeführt. Der Nachteil gegenüber den anderen Verfahren ist hierbei, dass das Betriebssystem bzw der Kernel entsprechend angepasst werden muss.

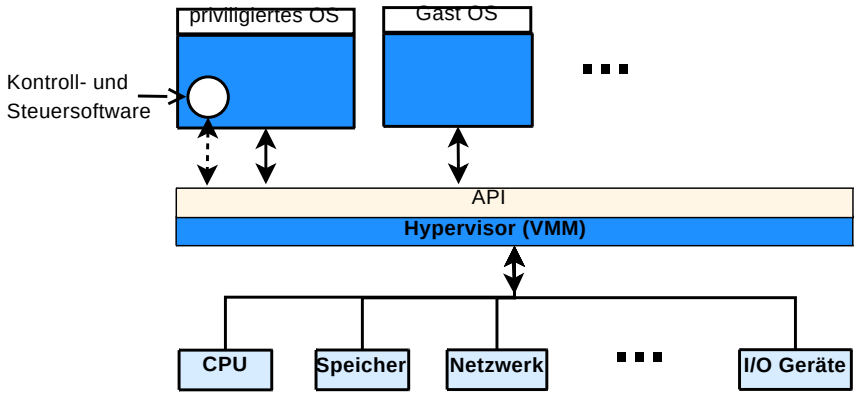


Abbildung 4.6: Aufbau eines Systems mit Paravirtualisierung.

Neben der Fallunterscheidung nach der Methode kann man Virtualisierungstechniken auch in die beiden Gruppen Hosted VM und Hypervisor VM unterteilen. Dabei bedeutet Hosted VM, dass das Gastsystem in einem anderen Betriebssystem ausgeführt wird, welches die Kontrolle über die Hardware besitzt. Bei einem Hypervisor VM gibt es einen Virtual Machine Monitor (VMM) [oder auch Hypervisor genannt] welcher die Ressourcen verwaltet und die Kontrolle über alle Gastsysteme inne hat. Die Gastsysteme setzen auf den VMM auf.
Wie anfangs erwähnt sind viele dieser Virtualisierungstechniken heute bereits im Einsatz, vornehmlich auf X86 basierenden Systemen. Eine Virtualisierung ist jedoch nicht ohne weiteres auf X86 Systemen möglich. Dies liegt an den Ringen bzw. Modi welche die X86 CPUs kennen. Jeder dieser Modi ist mit Rechten ausgestattet die den Zugriff auf bestimmte Speicherbereiche regeln. Befehl welche im Ring 0 ausgeführt werden dürfen uneingeschränkt auf alle Bereiche zugreifen. Das Betriebssystem bzw Kernel führen Anweisungen im Ring 0 aus. Anwenderprogramme benutzen den Ring 3 zur Ausführung, da sie weniger privilegiert sind wir die Betriebssystem Aufrufe. Falls eine Anwendung eine privilegierten Befehl ausführen möchte, so muss dieser an das Betriebssystem übergeben werden um im Ring 0 ausgeführt zu werden. Um mehrere Betriebssysteme auf dem selben System auszuführen müsste man den Ring 0 aufteilen. Dies würde unweigerlich zu Überschneidung und somit zu Fehlern führen. Bei der Virtualisierung löst man diese Probleme entweder durch Emulation von bestimmten Befehlen oder durch Anpassung des Betriebssystems unter Verwendung von Ring 1 und Ring 2 bei der Paravirtualisierung.
Im Bezug auf Selbstheilung bzw Self-X sei erwähnt, dass sich durch den Einsatz von Virtualisierung und speziell mit der Paravirtualisierung neue Möglichkeit ergeben. Da die VMM bei einer Paravirtualisierung die vollständige Kontrolle über das System besitzt, kann hier eine permanente Überwachungsfunktion eingebaut werden um gegebenenfalls gezielte Maßnahmen rechtzeitig einzuleiten. So könnten neben der passiven Komponenten durch Trennung der Instanzen auch eine aktive Komponente die Stabilität des gesamten

Systems weiter fördern. Momentan ist mir kein solches System auf dem Markt bekannt, weitere Arbeiten in der Forschung hierzu wird es in den nächsten Jahren geben.

4.5.2 real-world Scenarien / Sensornetze

Zur Verdeutlichung von Self-X Methoden in realen Anwendung möchte ich hier auf den drahtlosen Sensor Netzwerke (WSN) eingehen. WSN Netzwerken wird oftmals bescheinigt, dass sie das Potential für next generation industrielle Anwendungen haben. Momentan sind diese Netze aber eher in der Forschung und Entwicklung zu finden. Nur in wenigen Bereichen, wie etwa als Funksystem für die Abfrage von Wasserstand und Heizungsnutzung in Mietswohnung oder als Hilfssystem zur Raumtemperaturüberwachung findet man sie schon heute an. Zukünftig sollen durch WSN aber auch Bereiche wie etwa inventory management (item tracking) oder für sicherere Arbeitsumgebungen (track worker in hazardous areas) abgedecken.
Das Problem bei WSN Geräten ist, dass die Knoten (nodes) nur eine geringe Performance und Speicherkapazität bereitstellen, da das Hauptproblem die Energie darstellt. Die Geräte aber auch die eingesetzte Software, wie auch die verwendeten Netzwerkprotokolle sind im Hinblick auf Energiesparsamkeit optimiert. Dabei besteht immer ein Tradeoff zwischen Lebenszeit (lifetime) und dem QoS (quality of network service). Da die Geräte (node) oft unter harten Aussenbedingungen ausgesetzt sind, sind Fehlverhalten oder gar Ausfall von einzelnen oder mehreren Nodes möglich.
Das Fehler Management wird unterteilt in die Bereiche fault detection, diagnose und recovery, welche ich im Folgenden näher erklären werde. Es sei angemerkt, dass WSN Netze nicht mit herkömlichen Netzen völlig vergleichbar sind, z.b. ist es wichtig regionale Fehler zu erkennen anstelle von node-to-node Verbindungsfehlern.

Fault Detection

Die Detektion eines Ausfalls bzw einer Fehlfunktion ist die erste Phase des Fehler Managements. Es sollen unerwartete Fehler vom System identifiziert werden und die Information zur Diagnose weiter geleitet werden (value/sensor fault). Man unterscheidet dabei grundsätzlich zwischen offline und online Detektion, wobei online also real-time Detektion für WSN Systeme passender ist da WSN Geräte bauartbedingt bzw durch deren Einsatzgebiet bedingt häufiger ausfallen oder nicht erreichbar sein können.
Man unterteilt die Detektion ferner in expliziter und impliziter Detektion.

- Explizit durch Sensor Applikation, Alarm Meldung, Selbstdiagnose
- Implizit durch Management System, aktiv oder passiv (group detection vs hierarchical detection)
  - aktiv durch heartbeat Meldung (keep-alive)

– passiv model: event-driven, Alarm wenn Netzwerk- oder Nodeausfall identifiziert wurde, dabei muss die Datensenke die Überprüfung der Werte (value failure detection) übernehmen.

Existierende Ansätze findet man in zentralisierter oder verteilter Form, z.B. neighbor coordination, clustering approach, distributed detection. Ebenfalls denkbar sind Neuronalenetze und Fuzzy-Modelle für die Fehlerdetektion von Sensoren.

Fault Diagnose

In der Diagnose sind Fehler zu erkennen und von spontanen und irrelevanten Alarmmeldungen zu unterscheiden. Hierbei ist die Betrachtungsweise wichtig, z.B. Hardware-Fehler gegenüber von Software-Fehler, sowie die Ursache des erkannten Fehlers. Die Diagnose kann auf dem Knoten oder spätestends in der Datensenke stattfinden. Zur Feststellung der Korrektheit der Fehlerdiagnose werden Möglichkeiten des 'knowledge sharing' sowie anhand gespeicherter Datensätze verwendet. Der Einsatz von selbst-lernenden Systemen wird in der Forschung ebenfalls diskutiert.

Fault Recovery

Idealerweise wird in der 'recovery Phase' das Netz restrukturiert oder rekonfiguriert (ebenfalls die Software), so dass dem Ausfall eines oder mehrere Knoten entgegengetreten werden kann. Meistens erfolgt dieses durch Isolation von fehlerhaften Knoten auf Ebene der Netzwerkschicht. Es werden hierbei Schritte nötig wie etwa z.b. neuen Nachbarknoten für den Datentransfer zu bestimmen. Bei Ausfall eines cluster heads muss ein Neuaufbau des Clusters eingeleitet werden und ein neuer cluster head Knoten bestimmen werden. Datenverlust wird durch Redundanz begegnet. Proactives Vorgehen durch zusätzliche Abspeichern von Informationen anderer Knoten, jeder Knoten aktualisiert die Kopien mittels 'update message' Packeten. Nach einem detektierten Ausfall kann die Information aus den gespeicherten Informationen wiederhergestellt werden. Bedenkt man, dass gerade in WSN Netzwerken multi-hop Verbindung häufig eingesetzt werden, kann man somit Übertragungen einsparen und somit Energie sparen.
In einigen neueren Forschungsberichten wird der Einsatz von 'cooperative transmission in WSN' [9] untersucht. Hierbei wird neben dem Senden der eigenen Daten die zuvor übertragenen und empfangenen Daten einer Nachbarstation mittels des Code-Multiplex Verfahrens ebenfalls übertragen. Die Datensenke bekommt somit die Gelegenheit die Daten redundant also mehrfach zu erhalten und kann aus den möglicherweise unvollständigen Sendungen eine vollständige herausarbeiten ohne auf erneute Übertragung der Daten zu bestehen. Ob hierbei bei längerer Betrachtung ein Energiesparpotential einstellt ist noch nicht weiter untersucht worden, erscheint aber möglich. Diese recht einfache Methode zeigt, dass eine Selbst-Heilung bereits mit einfachen Redundanzen erreichbar ist.
Ein aktive Möglichkeit, welche in der WSN Forschung diskutiert wurde, ist der Einsatz eines beobachtenden Gateway Knotens. Angelehnt an das XCP Protokol kann der Gateway Knoten den Status des Netzwerkes anhand eines Mitschnitts beobachten und bei zu hohen

Paketverlusten die Anweisung geben die Datenrate/Packetrate mittels eines Feedbacks zu senken. Man muss hierbei bedenken, dass z.B. das TCP Protokoll aufgrund seines Overheads in WSN Netzen nicht eingesetzt wird. Es wird je nach Anwendungsfall meist ein speziell auf die Anwendung zugeschnittenes Protokoll verwendet. Durch Hinzufügen einer Entscheidungskomponente kann entsprechenden Netzwerküberlastungen entgegengetreten werden.

Ein robustes MAC Protokoll - LEACH

Gerade in Sensornetzen ist die Verwendung von speziell auf die Umgebung des Netzes angepasste Netzprotokolle von hoher Bedeutung. Bei den Protokollen des MAC sublayer (OSI Schicht 2) für den Einsatz in WSN Netzen sind zwei Voraussetzungen zu erfüllen. Zum einen muß das Protokoll hinsichtlich Ressourcen-Verbrauch optimiert sein und zum anderen dürfen Nachrichtenverluste durch Übertragungsprobleme, Ausfälle von Knoten usw. das Protokoll nicht sprichwörtlich in die Knie zwingen.
Eine Kategorie von Protokollen, welche in diesem Zusammenhang in der Forschung oft genannt wird, sind die 'schedule-based' Protokolle. Der Vorteil dieser Protokolle ist, dass die Verwendung des Sendeplan (transmission schedule) für die Vergabe der Sendezeiten dafür sorgt, dass Paket-Kollisionen bei den Empfängern nicht enstehen können und somit keine Mechanismen für das versteckte Sender Problem (hidden terminal problem) im Protokoll benötigt werden. Ein weiterer Vorteil ist, dass die beteiligten Knoten aufgrund ihrer genau festgelegten Sende- und Empfangsfenster - TDMA Schemata - lange Schlafphasen einplanen können. Nachteilig ist zu nennen, dass die Verteilung und der Aufbau des Sendeplan Ressourcen bindet. Durch den Einsatz von TDMA bedingt, müssen Taktung und Sendefenster-Überlappungen beachtet werden. Die Synchronization der Knoten untereinander ist hierbei sehr von Bedeutung. Zusätzlich ist zu beachten, dass die Kommunikation im gesamten Netz leidet wenn nur ein Knoten einen falschen Sendeplan abarbeitet. Die Fähigkeit das Netz in solch einen Fall neu zu konfigurieren ist nötig.
Ein Vertreter der schedule-based Protokolle mit Selbst-Konfigurations und Selbst-Heilungs Fähigkeiten ist das LEACH (Low-energy Adaptive Clustering Hierarchy) [12] Protokoll. In LEACH ist ein TDMA basiertes MAC Protokoll welches eine Cluster Technik benutzt. Die Knoten des Netzes werden in Cluster aufgeteilt und jeweils ein Knoten des Clusters als Clusterhead Knoten bestimmt. Dieser muss den TDMA Zeitplan (Sendeplan) koordinieren und an die anderen Cluster-Knoten weiterleiten. Der TDMA Zeitplan legt fest, wann welcher Knoten senden darf. Da das LEACH Protokoll für Anwendungen ausgelegt ist, wo Sensor-Knoten ein Ereigniss einer Datensenke mitteilen möchten und peer-to-peer Kommunikation nicht vorgesehen ist, können alle Knoten bis auf den Clusterhead und der sendende Knoten in dem Zeitfenster zumindest ihre Empfangseinheit ausschalten und somit Energie einsparen. Der Clusterhead nimmt die Nachricht des Knotens entgegen und leitet die Nachricht anschliessen an die Datensenke weiter. Der Clusterhead sammelt allerdings zunächst alle gesendeten Nachrichten um nur eine Sendung durchführen zu müssen. Dabei muss beachtet werden, dass die Kommunikation zwischen Knoten und Clusterhead Knoten viel günstiger ist wie das Versenden von Nachrichten vom Clusterhead Knoten zur Datensenke. Da der Clusterhead Knoten sehr Ressourcen fordernde Aufgaben wahrnimmt - Kommunikation mit Datensenke, Koordination des Clusters, permanentes Empfangen von Nachrichten - kann die Funktion des Clusterheads auf Dauer nicht ein Knoten alleine

bewältigen, seine Energie-Ressourcen würden zu schnell verbraucht sein und der Cluster wäre 'Kopf los'. Daher sieht das Leach Protokoll vor, dass die Role des Clusterhead Knotens nach Ablauf einer definierten Zeit oder bei Ausfall des Knotens einem anderen Knoten übertragen wird. Alle Knoten entscheiden für sich, ob sie die Role des Clusterheads übernemmen wollen oder nicht. Für die Entscheidung ist Maßgeblich entscheidend, wie lange der Zeitpunkt zurückliegt, wo der Knoten die Role inne hatte. Alle Knoten die nicht Clusterhead geworden sind suchen sich einen neuen Clusterhead Knoten anhand der Signalstärke Messung. Um die Cluster, besonders die Cluster in der Nachbarschaft, nicht zu stören, sucht sich jeder Clusterhead Knoten eine zufällig gewählten CDMA Code aus. Dieser Code wird an alle zugehörigen Cluster Knoten gesendet. Somit wird vermieden, dass ein Knoten am Rand des Clusters eine Übertragung eines anderen Clusters stört. Die Schwierigkeit des LEACH Protokolls ist es, eine gute Konfiguration für die Bestimmung des prozentualen Anteils an Clusterhead Knoten zu finden. Diese ist Abhängig von der Umgebung und der Anzahl der verwendeten Knoten.
Im Bezug auf Selbst-Heilung ist dieses Protokoll durch die Neuvergabe der Management Role (Clusterhead) interessant. Eine Störung der Kommunikation oder der Ausfall des Clusterhead Knotens wird spätestens nach der automatischen Neukonfiguration der Cluster begegnet.

4.6 Zusammenfassung und Ausblick

Durch den Einsatz von Methoden des Autonomic Computing können Systeme hinsichtlich Stabilität und Zuverlässigkeit bereits jetzt profitieren. In Zukunft wird der Aspekt Sicherheit durch den Einsatz von Selbstheilungs- und Selbstschutzmethoden hinzukommen. Gerade im Bereich der Betriebssysteme wird aufgrund der stark zunehmenden Komplexität der Systeme das Konzept des Autonomic Computing an Bedeutung gewinnen. Auf der einen Seite sind komplexe Systeme wegen ihres komplexen Aufbaues und der Mächtigkeit an Programmzeilen nur mit hohem Aufwand frei von Fehlern und auf der anderen Seite werden die Systeme immer anfälliger für Angriffe. Die in diesem Seminarbeitrag gezeigten Überlegungen hinsichtlich eines Einsatzes von Selbstheilungs- und Selbstschutzmethoden verdeutlichen, dass je nach geforderten Automatisierungsgrad eine Verlagerung der Komplexität von der Administration und Wartung des Systems hin zur Entwicklung stattfindet. Umso autonomer das System ausgelegt wird, desto komplexer wird die Entwicklung des Systems. Hinsichtlich der Betriebssysteme ist daher eine vollständige Automatisierung momentan nur schwer vorstellbar, jedoch sind teil-autonome Systeme bereits in der Erforschung. Neue Methoden wie selbst-lernende Systeme werden hier in naher Zukunft sicherlich ihren Beitrag leisten.
Neben den Betriebssystemen sind auch Rechnernetze durch ihre zunehmenden komplexen Strukturen bzw. im Falle von drahtlosen Sensornetzen aufgrund ihrer Eigenschaften vielfach auf Methoden des Autonomic Computing angewiesen. Gerade bei hohen Ausfallraten sind bereits in der Entwicklung und Konzeption von Netzprotokollen self-management Aspekte zu beachten. Bei den drahtlosen Sensornetzen hat dieses zu bereits verfügbare robuste Protokolle geführt. Jedoch sind diese Protokolle meist auf eine bestimmte Aufgabe oder Applikation des Netzes abgestimmt, somit ist eine einfache Adaption auf andere Netze meist nicht möglich.

Die Forschung im Bereich Betriebssysteme, Rechnernetze aber auch andere Gebiete wird in Zukunft auf Methoden des Autonomic Computing verstärkt zugreifen müssen. Dabei ist jedoch ein Grundsatz zu befolgen: *" Don't shoot in your own foot"*. Ein voll autonomes System benötigt keine Administration. Es stellt sich daher die Frage wer kontrolliert nun das System, wollen wir wirklich so ein System? Ein wünschenwertes System soll sicherlich möglichst autonom betrieben werden können, jedoch muss es weiterhin kontrollierbar bleiben. Die Administration und Wartung der Systeme soll reduziert aber nicht wegrationalsiert werden.

Literaturverzeichnis

[1] Swift, Anamalai, Bershad, Levy. *Recovering Device Drivers*, OSID 2004

[2] Koopman, *Elemets of the Self-Healing System Problem Space*, ICSE WADS 2003

[3] IBM, *Autonomic Computing: IBM's Perspective on the Sate of Information Technology*, 2001

[4] Yu, Mokhtar, Merabti, *Fault Managment in Wirless Sensor Networks*, IEEE Wireless Communications 2007

[5] Wang, Li, Zhang, *Efficient Self Protection Algorithms for Wireless Sensor Networks*, IEEE ICDS 2007

[6] Sheng, Chan, Xiangjun, Xianzhong, *Agent-Based Self-Healing Protection System*, IEEE Transactions on Power Delivery, Vol.21, No.2, 2006

[7] Tanenbaum, *Minix 3*, http://www.minix3.org

[8] Woldegebreal, Karl, *Network-Coding-Based Cooperative Transmission in Wireless Sensor Networks: Diversity-Multiplexing Tradeoff and Covereage Area Extension*, EWSN 2008

[9] Karl, Willig, *Protocols and Architectures for Wireless Sensor Networks, Wiley (2005)*

[10] Swift, Annamalau, Bershad, Levy, *Recovering Device Driver*, ACM Transactions on Computer systems, 2006

[11] Swift, Marting, Levy, Eggers, *Nooks: an architecture for reliable device drivers, 10. ACM SIGOPS European Workshop, 2002*

[12] Heinzelman, Chandrakasan, Balakrishnan, *An Application-Specific Protocol Architecture for Wireless Microsensor Networks*, IEEE Transactions on Wireless Networking, 2002