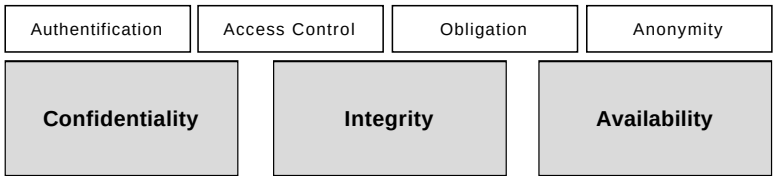
Zusammenfassung "Sicherheit in Daten- und Telekommunikationsnetzwerken" bei Dr. Posegga (SS 2000)

Michael Jaeger

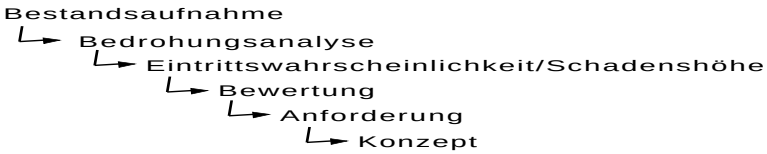
4.10.2000

Grundlagen

Die drei Grundpfeiler der Sicherheit sind *Confidentiality, Integrity* und *Availability.* Auf diesen aufbauend werden *Authentifikation, Zugriffskontrolle, Verbindlichkeit* und *Anonymität* gewährleistet:



Tanenbaum spricht von der *Geheimhaltung, Authentifikation, Nichtanerkennung* und *Integrität.* Um nun ein Sicherheitssystems zu planen, sollte man zunächst eine *Risikoanalyse* vornehmen, die ungefähr so beschrieben werden kann:



Sicherheit zieht sich durch alle Schichten des Netzwerkmodells:

- *Bitübertragungsschicht:* Das Anzapfen von Leitungen kann z.B. durch den Einsatz von Glasfaser oder mittels Gas gesicherter Kabelkanäle
- *Sicherungsschicht:* Hier kann bereits ein Verschlüsselungsalgorithmus eingesetzt werden, da jeder Router das Paket entpackt, bevor er es weiterschickt, und somit auch lesen kann.
- *Vermittlungsschicht:* Um sich vor Paketen von Eindringlingen zu schützen kann man z.B. eine Firewall installieren.
- *Transportschicht:* Eine Verschlüsselung auf dieser Schicht ist sehr sinnvoll und wird bereits häufig eingesetzt, jedoch eignet sich diese Schicht nicht für die *Authentifikation* und *Nichtanerkennung.*

Bei er Entwicklung von Kryptoalgorithmen sollte darauf geachtet werden, dass

- ▷ keine Nachrichten wiederholt werden können (z.B. durch einen Zeitstempel)
- ▷ das Maß an Redundanz sinnvoll gewählt wird (um die Erzeugung gültiger Nachrichten zu erschweren, aber auchdas erkennen von solchen nicht zu sehr zu erleichtern)

Heutzutage versucht man die Agorithmen so komplex wie möglich zu machen, während die Schlüssel möglichst kurz sind (früher war das umgekehrt). Dabei bestehen die meisten Operationen aus *Subsitutionen* und *Permutationen.*

Zugriffsorganisation

In Betriebssystemen sind zwei Arten des Zugriffsschutzes gängig:

ACLs überprüfen die Authentizität und die Authorisierung, indem in der *access control matrix*, die in einer sog. *access control list* implementiert ist, die Berechtigungen überprüft werden. Zu jedem Objekt gehört eine solche ACL, in der die Rechte für dieses Objekt definiert sind.

Capabilities Der Benutzer bekommt ein sog. *ticket* zugewiesen, in dem für seine Zugriffsrechte für Objekte im System gespeichert sind. Dies hat Performance-Vorteile und zudem können Berechtigungen kopiert und weitergereicht werden.

Im Gegensatz zum *late binding* bei ACL's werden Zugriffsentscheidungen bei Capabilities zum Zeitpunkt der Ticket-Vergabe getroffen. Aus diesem Grund ist das Zurückziehen von Rechten bei Capabilities höchstens über ein *expiry date* möglich. In der Praxis werden häufig hybride Ansätze verwendet, indem Capabilites aufgrund von ACL's vergeben werden (z.B. file-descriptors).

Kryptoalgorithmen

Heutzutage wird bei der Entwicklung neuer Kryptoalgorithmen das *Kerckhoff-Prinzip* angewandt, wonach der Algorithmus öffentlich gemacht wird und nur mit geheimen Schlüsseln gearbeitet wird. Dieses Verfahren hat sich in der Praxis als wesentlich sicherer erwiesen als die bisherige Vorgehensweise der *security by obscurity.* Als zusätzliche Anforderung an moderne Kryptolgorithmen stellt man heutzutage die Forderung nach *Rauschfreiheit*, also dass die Ausführungsgeschwindigkeit unabhängig von Schlüssel, Klartext bzw. Schlüsseltext sein soll.

Bei Kryptoalgorithmen werden im groben drei Klassen unterschieden: die symmetrischen, die asymmetrischen und diejenigen für Signaturen.

Symmetrische Algorithmen

DES

Verschlüsselungsstandard, 56 Bit, Blockcipher, Diffusion & Konfusion, Hardware-optimiert, CBC

Gechichte: Der wohl bekannteste Vertreter der Klasse der symmetrischen Algorithmen ist der *Data Encryption Standard*, ein von der NSA in Auftrag gegebener und von der IBM 1970 entwickelter Algorithmus.

Verfahren: DES ist ein Block-Algorithmus, der immer nur 64-Bit-Blöcke verschlüsselt. Klar- und Chiffretext haben die gleiche Länge und als Schlüssel kommt eine 64-Bit große Zahl zum Tragen, von der allerdings nur 56 Bit verwendet werden, da der Rest für Prüfsummen benötigt wird. Mathematisch besteht der DES nur aux XOR-Operationen, Permutationen und Substitutionen und verfolgt als Entwicklungskriterien die *Diffusion* (jedes Bit des Klartextes

und des Schlüssels soll möglichst viele Bits des Chiffretextes beeinflussen) und die *Konfusion* (die Statistik des Klartextes soll die des Chiffretextes derart komplex beeinflussen, dass ein Angreifer keine Vorteile daraus ziehen kann). DES wurde mit dem Ziel entwickelt, besonders gut in Hardware implementiert werden zu können.

Problem: Die Beschränkung der Schlüssellänge auf 56 Bit, die zu Anfang noch niemanden störte, wurde im Laufe der Zeit zu einem immer grösseren Problem, da es bei der heutigen Rechnerleistung immer schneller möglich ist, die Verschlüsselung mittels diverser Attacken zu brechen. Die Zielsetzung der guten Implementierbarkeit in Hardware hatte außerdem zur Folge, dass Realisierungen in Software performance-technisch nicht optimal sind. Als Folge ergaben sich die Entwicklungen von Double- und Triple-DES.
Wenn die Struktur und der Inhalt der Klartextes bekannt ist und ersterer günstig ist (8 Byte lang), dann kann durch einfache Zeichenersetzung eine Nschricht manipuliert werden (z.B. Datensätze einer Datei, die Jahresboni enthält [Tanenbaum]). Aus diesem Grund wird bei DES das *cipher block chaining (CBC)* angewandt, dass jeden Chiffreblock vor der Verschlüsselung mit dem vorigen Chiffreblock XOR-verknüpft. Der erste Block wird mit einem zufälligen Initialisierungsvektor XOR-verknüpft. Dieses Vorgehen erschwert die Kryptoanalyse sehr. Da bei dieser Methode aber zuerst ein 64 Bit-Block ankommen muß (mit Initialisierungsvektor), wird für interaktive Anwendungen der *cipher feedback mode (CFM)*, der bei jedem neu hinzukommenden Byte den bisherigen Schlüssel verwendet, dann mit dem neuen Byte XOR-verschlüsselt und schließlich das linke Byte schickt.

Double-/Triple-DES

Sicherer als DES, meet-in-the-middle attack bei 2DES

Geschichte: Aufgrund von Unzulänglichkeiten vom DES bzgl. der Schlüssellänge wurde ein Weg gesucht, den DES-Algorithmus ohne Änderungen zu grösseren Schlüssellängen zu verhelfen.

Verfahren: Anstatt wie beim DES nur mit einem Schlüssel, wird beim Triple-DES mit zwei Schlüssel k_1 und k_2 gearbeitet. Dabei wird der Klartext erst mit k_1 verschlüsselt, dann mit k_2 entschlüsselt und zuletzt wieder mit k_1 verschlüsselt: $c = e(k_1, d(k_2, e(k_1, m)))$. Dadurch erreicht man eine maximale Schlüssellänge von $2 \cdot 56$ Bit = 112 Bit (man könnte auch drei Schlüssel nehmen, was noch sicherer wäre, aber mehr Schlüsselverwaltungsaufwand mit sich bringen würde). Beim sog. **Double-DES** werden auch zwei Schlüssel verwendet, mit denen der Klartext zwimal verschlüsselt wird: $c = e(k_2, e(k_1, m))$.

Problem: Der Double-DES läßt sich mit Hilfe der sog. *meet-in-the-middle-attack* angreifen (da gilt: $c = e(k_2, e(k_1, m)) \Leftrightarrow d(k_2, c) = e(k_1, m)$). Ansonsten stellt dieses Verfahren eine effektive Erhöhung der Sicherheit dar und wird sehr häufig eingesetzt - im Gegensatz zu DES, da als nicht mehr sicher gilt.

IDEA

Bekannt durch PGP, Blockcipher, Schlüssel 128 Bit

Geschichte: Der *International Data Encryption Algorithm* wurde vor allem wegen seiner Verbreitung als Bestandteil von PGP bekannt. Bei der Entwicklung galt DES als Vorbild, die Schlüssellänge wurde jedoch von Anfang an mit 128 Bit sehr groß gewählt. Da er nicht in den USA entwickelt wurde, hatte er es allerdings sehr schwer, sich gegen DES durchzusetzen, dafür war er aber auf eine Implementierung in Software optimiert, was ihn auf Computern schneller als DES macht.

Verfahren: Wie beim DES erfolgt die Entschlüsselung mit dem gleichen Verfahren wie die Verschlüsselung, allerdings mit umgekehrter Reihenfolge der Teilschlüssel. Wie auch beim DES werden Blöcke der Länge 64 Bit chiffriert. Die maximale Schlüssellänge beträgt 128 Bit. Bei der Verschlüsselung hängt nach einer Permutation jedes Ausgabebit von jedem Eingabebit ab. Eine Verwendung des CFM-Mode ist möglich.

Problem: Bis heute sind keine Schwachstellen in diesem Algorithmus aufgetaucht und er gilt als die große Nummer zwei hinter DES als meistbenutzter symmtrischer Verschlüsselungsalgorithmus.

RC4

Geschichte: Das von Ron Rivest entwickelte RC4-Verfahren wird heute auch sehr viel verwendet. Ursprünglich nicht im Source-Code vorliegend gelang dieser durch unbekannte Kanäle an die Öffentlichkeit, und wurde weiter geprüft - ohne "Erfolg".

Strom-Chiffre, Pseudo-Zufallsgenerator, One-Time-Pad, Schlüssellänge variabel

Verfahren: Im Gegensatz zu DES und IDEA und den meisten anderen symmetrischen Algorithmen handelt es sich bei RC4 nicht um einen Block-, sondern um einen *Strom-Chiffre*. Dies ist ein Pseudo-Zufallsgenerator, der eine Zufallsfolge für ein One-Time-Pad liefert. Der Schlüssel liefert dabei den Anfangswert für den Pseudozufallsgenerator. Die Länge des Schlüssels kann variabel gewählt werden, was ihn sehr zukunftssicher macht.

Problem: Bis heute sind keine nennenswerten Schwächen bekannt geworden, und RC4 gilt als große Konkurrenz für DES und IDEA.

RC5

Verfahren: Im Gegensatz zu RC4 ist dieser Algorithmus ein Blockchiffre, bei dem alle variabel ist: Schlüssellänge, Blocklänge und Rundenanzahl.

Alles variabel (Schlüssellänge Blocklänge, Rundenanzahl)

Blowfish

Verfahren: Bruce Schneier entwickelte diesen Algorithmus als Blockchiffre mit 64 Bit Blocklänge, ähnlich DES. Die Schlüssellänge ist jedoch variabel (max. 448 Bit).

Blockchiffre, Schlüssellänge variabel bis 448 Bit

Asymmetrische Algorithmen

Diffie-Hellman

Geschichte: Da dieser Algorithmus ursprünglich zum Schlüsselaustausch entworfen wurde, wird er auch *Diffie-Hellman-Schlüsselaustausch-Algorithmus* genannt. Mit diesem sehr einfachen Verfahren begründeten Diffie und Hellman die Public-Key-Kryptografie.

Schlüsselaustauschverfahr diskreter Logarithmus, Schlüssellänge ab 786 Bit

Verfahren: A und B einigen sich auf ein $n \in \mathbb{P}$ und ein $g < n$, $g \in \mathbb{N}$. A erfindet nun x, B erfindet y wobei $x, y < n$. Der Schlüsselaustusch funktioniert nun wie folgt:

$$A \rightarrow \text{liefert } (a = g^x \mod n) \text{ an} \rightarrow B \rightarrow \text{liefert } (b = g^y \mod n) \text{ an} \rightarrow A$$

A berechnet sich nun aus b die Zahl $k_A := b^x \mod n$ und B berechnet sich $k_B := a^y \mod n$. Interessanterweise ist nun aber $k_A = k_b =: k$ - A und B haben einen gemeinsamen Schlüssel erzeugt!

Ein Angreifer müsste nun, da k niemals über die Leitung geht, den diskreten Logarithmus durchführen, um an den Schlüssel heranzukommen, was nach wie vor sehr schwer ist. Die Schlüssellänge kann frei gewählt werden.

Problem: Da eine Berechnung des diskreten Logarithmus effektiver ist, als eine Brute-Force-Attacke, müssen die Schlüssel deutlich länger sein als die von DES oder IDEA (mindestens 768 Bit für alle k).

RSA

Asymmetrische Verschlüsselung+Schlüsselaustausch, patentiert bis Sept. 2000, 1000-mal langsamer als symmetrische

Geschichte: Der wohl bekannteste aller public-key-Verfahren ist das von Rivest, Shamir und Adleman entwickelte. Es kann im Gegensatz zu Diffie-Hellman nicht nur zum Schlüsselaustausch, sondern auch zum Verschlüsseln verwendet werden. Es basiert ebenfalls auf dem Problem des diskreten Logarithmus.

Varfahren: Man geht dabei wie folgt vor:

$$\text{Öffentlich}: \; n = p \cdot q, \quad p, q \in \mathbb{P}, \quad e \nmid \varphi(n), \, e \in \mathbb{N}$$

$$\text{Geheim}: \; d = e^{-1} \mod \varphi(n)$$

$$B: \; c = m^e \mod n \quad , \quad A: \; m = c^d \mod n = m^{e \cdot e^{-1} \mod \varphi(n)}$$

e entspricht in diesem Fall dem öffentlichen Schlüssel von A, während d der geheime Schlüssel ist.

Problem: Public-key-Verfahren sind etwa 1000-mal langsamer als symmetrische, da diese in der Regel ausschließlich einfache, mathematisch anspruchslose Funktionen verwenden. Außerdem sind asymmetrische Verschlüsselungsverfahren in der Implementierung wesentlich komplexer als symmetrische, wodurch sich leichter Fehler einschleichen können. Aus diesem Grund wendet man in der Praxis häufig sog. *hybride Verfahren* an, bei denen asymmetrische Verfahren lediglich zum Austausch der symmetrischen Schlüssel verwendet werden. Beim RSA-Verfahren kommt noch speziell hinzu, dass es bis Mitte September 2000 patentiert war und so nur gegen Lizenzgebühren verwendet werden durfte, was seinem Einsatz in frei verfügbaren Produkten hinderlich war.

Elliptische Kurven

Nur Schlüsselaustausch oder Verschlüsselung, 10-mal schneller als RSA, relativ jung, Smartcards

Geschichte: Dieses noch sehr junge Verfahren kann sowohl zur Signatur, als auch zur Verschlüsselung und zum Schlüsseltausch verwendet werden. Aufgrund seiner mathematischen Grundlagen ist dieses Verfahren etwa zehnmal schneller als RSA und benötigt bei gleicher Sicherheit wesentlich kürzere Schlüssel als letzterer. Auch die Signatur ist wesentlich kürzer als beim RSA. Diese Eigenschaften prädestinieren diesen Algorithmus für den Einsatz in Chipkarten, die nur über wenig Rechenleistung und wenig Speicher verfügen. Außerdem können Verfahren auf ECCs entweder nur zur Signatur oder nur zur Verschlüsselung verwendet werden.

Verfahren: Die mathematische Grundlage stellt die Körpertheorie. Eine elliptische Kurve definiert nun eine Kurve auf einem Körper, die die interessante Eigenschaft hat, dass jede Gerade, die die Kurve schneidet dies in genau drei Punkten tut (außer der Geraden parallel zur y-Achse und der, die die Kurve nur berührt).

Problem: Da das Verfahren noch relativ jung ist, wird ihm noch nicht das gleiche Vertrauen entgegengebracht wie RSA. Bis heute wurden jedoch noch keine Schwächen in diesem Algorithmus entdeckt.

Signaturen

Unter einer digitalen Signatur kann man sich das Pendant zur Unterschrift im "richtigen Leben" vorstellen. Ziel bei der Entwicklung von Signatur-Algorithmen ist es, sie schwer fälschbar, leicht überprüfbar und das signierte Dokument schwer veränderbar zu machen. Dabei ist es wichtig, dass eine Signatur nur mit einem geheimen Schlüssel erstellbar ist, während die Prüfung, ob eine Signatur korrekt ist, auch ohne Kenntnis des geheimen Schlüssels möglich sein sollte.

RSA

Signatur mittels *private-key* Verschlüsselung

Geschichte: Wie bereits oben erwähnt kann man auch mittels RSA Signaturen erstellen - wie mit eigentlich allen Public-Key Verfahren.

Verfahren: A verschlüsselt die zu signierende Nachricht mit seinem geheimen Schlüssel. Die asymmetrische Eigenschaft des RSA gewährleistet nun, dass das Dokument nur mit dem Gegenstück des geheimen Schlüssels von A, nämlich seinem öffentlichen Schlüssel dechiffriert werden kann. Eine Veränderung der signierten Nachricht würde starke Konsequenzen auf den entschlüsselten Text haben.

Problem: Die Kritikpunkte sind die gleichen wie beim RSA-Algorithmus selbst: er ist langsam und es gibt allgemein nur wenige asymmetrische Kryptoalgorithmen zum Verschlüsseln/Signieren.

Diskreter Logarithmus

El Gamal, DSS, nur Signatur möglich, längere Schlüssel als RSA, dafür schneller bei gleichlangen Schlüsseln

Geschichte: Die bekanntesten zwei Signatur-Algorithmen auf Basis des diskreten Logarithmus sind der *El Gamal-Algorithmus* und der *Digital Signature Standard (DSS)*. Da man das Verfahren vielseitig variieren kann, gibt es sehr viele Signaturverfahren, denen dieses Prinzip zu Grunde liegt.
Weil diese Algorithmen nur zur Signatur verwendet werden können, lassen sie sich nicht zur Verschlüsselung "mißbrauchen", was sie politisch unbedenklich macht. Außerdem unterliegen sie im Gegensatz zum RSA-Algorithmus keinem Patent und können frei genutzt werden.

Verfahren: Sei $p \in \mathbb{P}$ und $g < p$ mit $g \nmid p$. A denkt sich nun einen geheimen Schlüssel $x < p$ aus und berechnet den öffentlichen Schlüssel $g^x \mod p$, den B bekommt. Wenn A nun eine Nachricht m signieren möchte, dann berechnet A eine Zufallszahl y mit $y \nmid (p-1)$ und berechnet g^y und $m = x^* g^y + y^* s \mod (p-1)$, wobei s und g^y zusammen die digitale Unterschrift für m bilden. B prüft nun die Signatur ohne Wisse nvon x und y durch

$$g^m = g^{x^* g^y + y^* s} \mod p = g^{x^* g^y} \cdot g^{y^* s} \mod p$$

Problem: Während die Erstellung der Signatur bei diesem Verfahren etwa gleich schnell geht wie beim RSA, ist letzterer bei kleinen Schlüssellängen etwa 10-mal schneller, dafür muß bei gleicher Sicherheit die Länge des Schlüssels beim RSA immer etwas größer sein. Die Tatsache, dass für jede Signatur ein neuer Zufallswert generiert werden muß ist ein nicht zu unterschätzender Nachteil der Arbeit mit diskreten Logarithmen, da die Erzeugung guter Zufallszahlen als sehr schwer gilt.

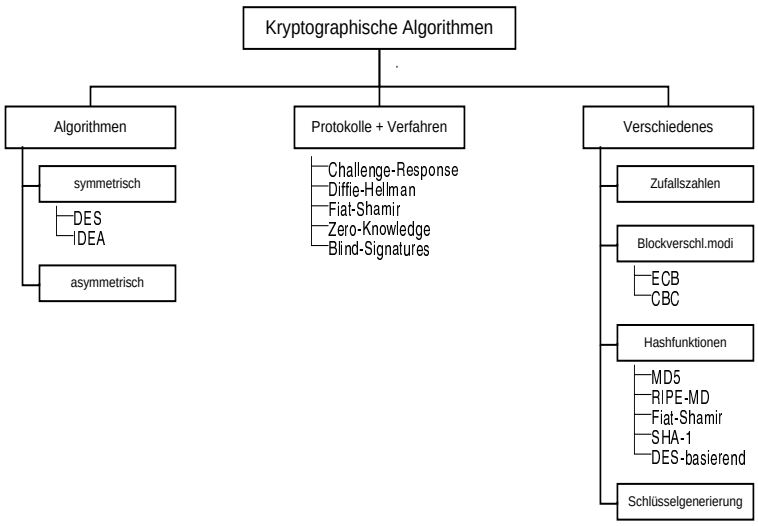


Abbildung 1: Kryptographische Techniken

Hashfunktionen

Falltürfunktion, Komprimierung, MD4 (512 → 128, schnell), SHA (512 → 160), MD5 (bereits angeschlagen), RIPEMDS-160 (Europa)

Geschichte: Bei Hashfunktionen handelt es sich um Falltürfunktionen, die eine sehr großen Wert auf auf einen relativ kleinen komprimieren, ohne dass eine Kollision des Hashwerts zweier Nachrichten wahrscheinlich und das Finden einer Kollision sehr schwer ist. Außerdem können Hashwerte auch als Pseudozufallsgeneratoren und zur Signatur eingesetzt werden, da die Echtheit ohne einen geheimen Schlüssel überprüft werden kann.
Besonders bekannte Vertreter dieser Sorte von Signaturalgorithmen sind *MD4, MD5* und *SDA*.

Verfahren: Bei *MD4* werden Blöcke der Länge 512 auf Blöcke der Länge 128 komprimiert. *MD5* als Nachfolger von MD4 generiert ebenfalls 128 Bit-Hashes aus 512 Bit-Blöcken und ist heute sehr verbreitet.
Eine andere Weiterentwicklung von MD4 ist *SHA (Secure Hash Algorithm)*, der in den USA als Standard für die Signatur entwickelt wurde. Für einen Blöck der Länge 512 liefert es einen Hashwert der Länge 160 und gilt seit den aufgedeckten Schwächen der MDx-Algorithmen als die sicherste Hashfunktion. Die in Europa entwickelte Hashfunktion *RIPEMD-160* arbeitet ähnlich wie SHA und gilt als hoffnungsvoller Newcomer.

Problem: MD4 gilt schnell als unsicher, da sich das Finden von Kollisionen als nicht besonders schwer erwies. Auch das Image von MD5 ist heutzutage schon angeschlagen, da auch dort Möglichkeiten gefunden wurde, Kollisionen relativ einfach zu finden.

Um nun Signaturen einzusetzen, muß man eine Zertifizierungshierarchie aufbauen. Dabei sollte man sich an dem Standard `X.509` orientieren, der als Rahmenwerk zu verstehen ist. Leider sind die Produkte, die für die digitale Signatur vorhanden sind noch nicht gut genug auf den Endanwender abgestimmt, der Aufbau eine PKI noch zu kompliziert und die politischen Rahmenbedingungen noch zu ungeklärt, als dass sich die digitale Signatur bis heute auf breiter Basis durchgesetzt hat.

Eine Diagramm soll nun eine Überblick über die vorgestellten Verfahren geben:

Authentifikationsprotokolle

Diese Protokolle dienen lediglich dazu, eine *Echtheitsprüfung* durchzuführen, indem man

- einen *gemeinsamen geheimen Schlüssel* verwendet (dabei sendet A an B eine Zufallszahl, die B entschlüsselt und an A zurückschickt, der sie dann wieder mit dem Schlüssel verschlüsselt und so prüft, ob die Nachricht von A kam)
- einen *Schlüsselaustausch nach Diffie-Hellman* vornimmt
- einem *Schlüsselverteilzentrum* vertraut
- die *Kerberos-Methode* verwendet (setzt voraus, dass auf den Rechnern die Uhren synchron laufen; der Benutzer schickt den Namen im Klartext, der Server schickt Sitzungsschlüssel und Ticken mit Geheimschlüssel verschlüsselt zurück, mit dem Ticket, dem Servernamen und einem Zeitstempel verschlüsselt mit dem Sitzungsschlüssel kann sich der Benutzer nun Tickets von dem TGS für einen anderen Server holen, wobei auch die Kommunikation mit dme Ticket und einem Zeitstempel verschlüsselt wird)
- *öffentliche Verschlüsselung* einsetzt (A schickt B einen mit B's öffentlichem Schlüssel kodierten Sitzungsschlüssel; B schickt eine Zufallszahl mit dem Sitzungsschlüssel an A zurück, verschlüsselt mit A's ö. Schlüssel; A schickt nun die Zufallszahl von B mit dem Sitzungsschlüssel verschlüsselt an B zurück; A und B müssen dafür allerdings ihre ö. Schlüssel kennen)

Signaturen

Ziel der Verwendung von Signaturen ist, dass

- der Empfänger die Identität des Absenders überprüfen kann
- der Sender den Inhalt der Nachricht nicht verleugnen kann
- der Empfänger die Nachricht nicht selbst erzeugen kann

Mögliche Vorgehensweisen sind

- *Zentrale Zertifizierungsstelle*, der alle vertrauen, und die alle Nachrichten beglaubigt (aber auch lesen kann)
- Verschlüsselung mit dem öffentlichen Schlüssel von B und vorherige Verschlüsselung mit dem *geheimen Schlüssel* von A ($e(O_A, d(P_B, m))$). Probleme entstehen, wenn der geheime Schlüssel von A publik wird oder geändert werden soll
- die Verwendung eines *message digest* (damit werden 128 Bit Hashwerte ermittelt, die mit Hilfe eienr öffentlichen Hashfunktion erstellt werden). Das md kann dann auch noch mit privaten Schlüsseln signiert werden, was Zeit spart (Achtung: *birthday attack* zum Finden von Kollisionen)

Krypto-Attacken

Die momentan verbreitetsten Attacken gegen Krypto-Algorithmen sind:

cypher only attack: Der Angreifer kennt nur den Schlüsseltext und versucht anhand von diesem den Schlüssel oder den Klartext herauszufinden.

known plaintext attack: Der Angreifer kennt mehrere Klartext-Schlüsseltext-Paare und verfährt wie oben.

chosen plaintext/cyphertext attack: Der Angreifer kann sich beliebige Klar-Schlüsseltext-Paare generieren.

brute force attack: Der Angreifer versucht den Schlüssel durch systematisches, vollständiges Absuchen des Schlüsselraums zu finden.

Daneben gibt es natürlich noch Verfahren, die auf spezielle Eigenschaften der jeweiligen Algorithmen abgestimmt sind, wie die *man in the middle attack* bei Double DES.

Sicherheitsprotokolle

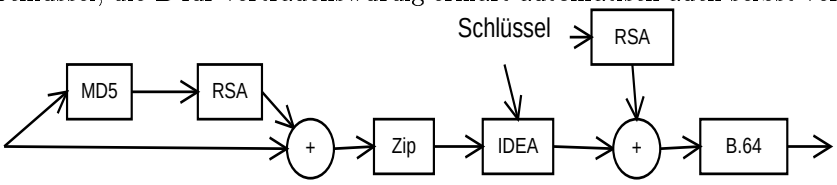
Spricht man von Sicherheit im Netzwerk, so werden im Allgemeinen verschiedene Schichten gemäß dem Schichtenmodell beim IP-Protokoll unterschieden:

- *Anwendung:* PGP, PEM, S/MIME, Basic Authentication, S-HTTP
- *Transport:* Socks, SSL
- *Netzwerk:* IPSEC
- *Sicherung:* PAP/CHAP, ECP, PPTP

Anwendungsschicht

PGP

Pretty Good Privacy ist eine Protokoll, das ursprünglich zur sicheren Verschlüsselung von E-Mails entwickelt wurde. Es kombiniert die Kryptoverfahren MD5, RIPEMD-160, SHA-1 zur Signierung und RSA, IDEA, Triple-DES und CAST zur eigentlichen Verschlüsselung. Zu guter letzt kommt noch ein Komprimierungsalgorithmus zum Einsatz. Durch die Möglichkeit der Signierung von Schlüsseln kann ein sog. *web of trust* aufgebaut werden, wo Person A z.B. Person B vertraut und alle Schlüssel, die B für vertrauenswürdig erklärt automatisch auch selbst vertraut.



PGP ist unterstützt die Schaffung denzentraler Vertrauensnetzwerken und deshalb für Firmen nicht so interessant, da diese eine hierarchische Schlüsselstruktur bevorzugen. Auch das Ungültigmachen von Zertifikaten ist sehr schwer, da ein Schlüssel nach dem Import erstmal im System solange gültig ist, bis er wieder gelöscht wird - und das auf jedem einzelnen Rechner. Auch die Tatsache, dass PGP keinem Standard entspricht, läßt viele Firmen vor desen Einsatz zurückschrecken, wobei ein Standardisierungsprozess gerade am Anlaufen ist. Bleiben am Ende noch die nicht vorhandene MIME- und Chipkarten-Unterstützung, die einer weiteren Verbreitung dieses Protokolls entgegen stehen.

Signatur: MD5, RIPEMD-160, SHA-1; Verschlüsselung: RSA, IDEA, 3DES, CAST; Signatur oder/und Verschlüsselung; Web-Of-Trust, kein Standard, kein MIME, Smartcards

All diesen Nachteilen steht die Tatsache gegenüber, dass PGP für Privatleute frei verfügbar ist und jedem Benutzer die Möglichkeit gibt, Daten derart sicher zu Verschlüsseln, dass es momentan selbst beim allergrößten Hardwareaufwand nicht möglich ist, die Verschlüsselung in angemessener Zeit zu brechen. Diese Tatsache

und die große Publicity, die mit dem Gerichtsprozess gegen den Erfinder von PGP einherging, führten zu einer großen Verbreitung dieses Protokolls bei der Verschlüsselung von E-Mails, aber auch binärer Daten.

PEM

Offizieller Standard; MD5, DES, RSA; Signatur & Verschlüsselung; MIME-support schlecht, 7-Bit-ASCII, Zertifizierungshierarchie

Speziell für die Verschlüsselung von E-Mails wurde seinerzeit das Privacy Enhanced Mail entwickelt. Nachrichten werden immer verschlüsselt und signiert, wobei die Algorithmen DES, RSA und MD5 zum Einsatz kommen. Damit gehört es zu der Klasse der hybriden Algorithmen.

Leider ist der Standard schon sehr alt und DES gilt mittlerweile nicht mehr als absolut sicher. Auch RSA wird nur bis zu einer Schlüssellänge von 1024 Bit unterstützt. MD5 entspricht ebenfalls nicht mehr dem Sicherheits-Standard von heute und die Verwendung von nur 7-Bit-ASCII macht eine vorherige Umwandlung von 8-Bit-ASCII nötig. Auch die Zusammenarbeit mit dem MIME-Standard ist ausgesprochen schlecht und Schlüssel müssen von einem Trust-Center verwaltet werden, die zu der Zeit, als PEM entwickelt wurde noch nicht im benötigten Anzahl gab. Aus diesen Gründen floppte PEM und spielte niemals eine wichtige Rolle im Krypto-Geschehen.

Vergleich PGP↔PEM

Merkmal	**PGP**	**PEM**
Verschlüsselung?	+	+
Authentifikation?	+	+
Nichtanerkennung?	+	+
Kompression?	+	-
Kanonische Umwandlung?	-	+
Mailinglisten?	-	+
base64-Kodierung?	+	+
Verschlüsselung?	IDEA	DES
Schlüssellänge	128	56
Alg. für Schlüsselmanagement?	RSA	RSA oder DES
Schlüssellänge für Schlüsselmngm.?	384/512/1024	variabel
Namensbereich	benutzerdef.	X.400
X.509-übereinstimmung?	-	+
Vertrauen nötig?	-	+
Schlüsselzertifizierung?	Ad-Hoc	IPRA/PCA/CA
Schlüsselwiderruf?	schlecht	besser
Nachrichten lesbar?	-	-
Unterschriften lesbar?	-	+
Internet-Standard?	-	+
Entwickler?	kleines Team	Normenausschuß

S/MIME

Große Verwbreitung durch Marktführer; Signatur: MD5, SHA; Verschlüsselung RC2 (40 Bit), DES, 3DES

Secure MIME wurde als Erweiterung des MIME-Standards zur Verschlüsselung von Nachrichten eingeführt. Die Signatur von Nachrichten ist im Gegensatz zu PEM optional. Die Verbreitung dieses Standards wurde vor allem durch eine Integration in die meist verbreiteten Mail-Programme von Netscape und Microsoft forciert. Als Algorithmus wird der RC2 mit 40 Bit Schlüssellänge verwendet, DES und Triple-DES sind optional. Zur Signatur werden MD5 und SHA verwendet, eine Zerti-

fizierungshierarchie ist möglich aber nicht obligat. Der Schlüsselaustausch erfolgt mittels RSA.
Aufgrund der weiten Verbreitung durch Standard-Mail-Programme gilt S/MIME als *der* kommende Verschlüsselungsstandard für E-Mails.

Mailtrust

PEM, unterstützt PSE, kein MIME, CA-Hierarchie

Mailtrust stellt eine Erweiterung von PEM dar und ist ein Standard des deutschen Industrieverbands Teletrust. Er verfügt über zusätzliche Nachrichtenformate und bessere kryptografische Verfahren. In dem Standard enthalten ist eine Schnittstelle zu einem *PSE (Personal Security Environment)* - im Regelfall eine Chipkarte. MIME-Unterstützung ist nicht vorgesehen, dafür entspricht aber die Unterstützung von Chipkarten und Zertifizierungshierarchien dem Signaturgesetz in Deutschland. Neuere Versionen von Mailtrust sollen auch S/MIME unterstützen, da sich PEM nicht richtig durchsetzen konnte.

Basic Authentication

einfacher HTTP-Mechanismus, unsicher

Dies ist der einzige Sicherheitsmechanismum, den HTTP seit der Version 1.0 integriert hat, weshalb ist es in eigentlich alle Browser integriert ist. Das Format, in dem das Paßwort übertragen wird ist zwar nicht für Menschen lesbar, jedoch gilt das verwendete Kryptoverfahren auch nicht als sicher.

S-HTTP

Betrifft nur HTTP

Um Netzwerkkommunikation, die über HTTP läuft zu sichern, wurde das SHTTP-Protokoll entworfen. Es funktioniert ähnlich wie SSL (so wird auch ein einmaliger Transaktionsschlüssel für jede Sitzung zufällig generiert), ist jedoch auf HTTP beschränkt und zudem proprietär. Aufgrund dieser Eigenschaften hat es sich nicht gegen SSL durchsetzen können und ist heutzutage kaum noch relevant.

Transport-Schicht

SSL

Handshake, asymmetrischer Austausch symmetrischer Sitzungsschlüssel, zwischen Transport und Anwendungsschicht, einfach Integration

Dieses Protokoll basiert auf einem Handshake-Protokoll zwischen Client und Server. Der Server besitzt einen öffentlichen Schlüssel, der von einer Zertifizierungsinstanz signiert sein kann. Diesen schickt er dem Client, damit der die Echtheit des Servers über die Signatur prüfen und einen zufälligen Sitzungsschlüssel generieren kann, der in der folgenden Session als symmetrischer Sitzungsschlüssel verwendet wird. Dies hat den Vorteil, dass jede Sitzung mit einem anderen Schlüssel abgehalten wird, die Authenzität des Server geprüft werden kann, und zur eigentlichen Verschlüsselung der Daten nicht ein aufwendige public-key-Verfahren, sondern ein schnelles symmetrisches verwendet werden kann. Da sich dieses Protokoll auf der Transport-Schicht abspielt, ist es für Anwendungen transparent und ermöglich so z.B. eine mittels SSL gesicherte Kommunikation zwischen einem FTP-Server und einem FTP-Client, ohne dass einer von beiden umprogrammiert werden müsste.

Netzwerk-Schicht

IPSEC

Vermittlungsschicht, Authentifizierung, Verschlüsselung, IKMP-Schlüsselmgmnt, IPv6-Bestandteil, auch mit UDP möglich

Wie der Name schon sagt, ist IPSEC ein Protokoll der Vermittlungsschicht und sichert die Kommunikation über IP. Hier besteht die Möglichkeit der Authentifikation über *authentication headers* und der symmetrischen Verschlüsselung von IP-Paketen. Das Schlüsselmangement wird getrennt von IPSEC über *IPSP* und

später über *IKMP* mittels asymmetrischer Verfahren vollzogen. IKMP funktioniert dabei etwa so wie SSL.

Da IPSEC fester Bestandteil von IPv6 sein wird, kann mit einer großen Verbreitung gerechnet werden.

Auf den ersten Bilck ähneln sich SSL und IPSEC sehr, da beide einen sicheren Tunnel unterhalb der Anwendungsebene bieten. SSL läßt aber nur Ende-zu-Ende-Verbindungen zu und ist näher an der Anwendung. Auch sind UDP-basierte Dienste mit SSL nicht möglich, dafür läßt es sich aber einfacher in bestehende Netze integrieren, da es keine Umstellung der Protokolle fordert.

Sicherungs-Schicht

CHAP

Ebene 2 unter IP, Hashwert mittels Zufallszahl, sicherer Kanal

Das zu PPP gehörige *Challenge Handshake Protocol* realisiert ein einfaches Challenge-Response-Verfahren auf der Ebene 2 noch unter IP. Eine Partei schickt der anderen dabei einen Zufallswert, den diese mit einer schlüsselabhängigen Hash-Funktion bearbeitet. Stimmt das Ergebnis mit dem der ersten überein, ist alles in Ordnung.

Diese Verfahren kommt ohne Public-Key-Verfahren aus, erfordert jedoch einen sicheren Kanal.

ECP

verschiedene Krypto-Verfahren auswählbar

Das *Encryption Control Protocol* sorgt dür die gesicherte Übertragung über PPP und löst CHAP ab. Im Gegensatz zu CHAP bietet ECP sogar die Möglichkeit, dass die Kommuikationspartner ein Verschlüsselungsverfahren beim Verbindungsaufbau aushandeln.

Smartcards

Ist die Rede von Chipkarten, so unterscheidet man in der Regel zwei Typen von Karten:

Speicherkarten: Sie verfügen nur über eine sehr beschränkte Funktionalität und werden in der Regel nur zum Speichern und Abrufen von Daten verwendet. Sie sind sehr billig in der Herstellung und eine intergrierte Sicherheitslogik schützt sie vor der Manipulation der Daten, so dass sie für den Einsatz als Wertkarten (z.B. Telefonkarten) und Ausweiskarten (z.B. Krankenkarte) geeignet sind.

Prozessorkarten: Besitzen eine relativ große Speicherkapazität und relativ viel Rechenpower, so dass sie Daten mittels Krypto-Algorithmen sichern können (z.B. Bankkarte, SIM). Für die schnellere Berechnung von Kryptofunktionen besitzen sie optional einen Coprozessor, der auf diese Aufgabe spezialisiert ist. Leider sind sie in der Herstellung recht teuer.

Die Datenübertragung von der Chipkarte zum Host kann *kontakbehaftet, kontaktlos* über Entfernungen bis zu einem Meter (was für elektronische Geldbörsen nicht unbedingt wünschenswert ist) oder *dual-interfaced* erfolgen.

Chipkarten-Betriebssysteme

Dateiübertragung, Ablaufsteuerung, Dateiverwaltung, Kryptographie

Zur Anfangszeit der Chipkartenentwicklung gab es nur Spezialprogramme für Chipkarten und keine allgemeinen Betriebssysteme, wie man sie von PC's her kennt. Alle Aufgaben wie die Ressourcenverwaltung, die normalerweise ein Betriebssystem übernimmt, mußten also jedesmal für jeden neuen Kartentyp neu entwickelt werden

(z.B. bei der SIM vom C-Netz). Heutzutage gibt es spezielle Chipkartenbetriebssysteme, die die Aufgaben *Dateiübertragung, Ablaufsteuerung, Dateiverwaltung, Verwaltung und Ausführung von Kryptoalgorithmen* übernehmen, so dass sich der Entwickler auf sein Programm konzentrieren kann. Dieses Vorgehen trägt sehr zur Sicherheit der Karte bei, da man sich auf bereits überprüfte Module verlassen kann und sich bei jeder Neuprogrammierung Fehler einschleichen können. Dies verkürzt den Entwicklungszyklus erheblich und spart Geld.

Die Zertifizierung von Betriebssystemen bieten große Anbieter wie die ITSEC an, da sie aber sehr teuer ist, wird diese Möglichkeit heutzutage nur durch wenige große Anbieter genutzt.

Im Folgenden sollen die wichtigsten Funktionen von Chipkartenbetriebssystemen erläutert werden:

Dateiverwaltung: Die Speicherverwaltung muß aufgrund des nur sehr begrenzt vorhandenem RAM-Speicher sehr effizient im Platzverbrauch und in der Geschwindigkeit sein. Das hat zur Folge, dass die Zugriffskontrolle nicht sehr umfangreich erfolgen kann und der Speicherplatz gelöschter Dateien u.U. nicht wiederverwendet werden kann, da ein Fragmentierung wie sie bei großen Dateisystemen bekannt ist aus Platzgründen vielleicht nicht möglich ist.

Dateien: Sie bestehen in der Regel aus einem *Dateiheader* für Attribute und dem *Dateibody* für die Daten. Es wird unterschieden zwischen *dedicated files (DF)* - den Verzeichnissen - und *elementary files (EF)* - den eigentlichen Dateien. Die Adressierung von Dateien erfolgt nun logisch über den Pfad und nicht direkt über eine Angabe des Speicherplatzes. *Transparente Dateien* haben dabei keine innere Struktur, während *linear fixed* eine Verkettung gleich langer Datensätze und *linear variable* eine Verkettung von Datensätzen individueller Länge beschreibt. Dateien des Typs *cyclic* bestehen aus records gleicher Länge, haben allerdings eine zirkuläre Struktur (nach dem Lesen des letzten Datensatzes ist man wieder beim ersten), ausführbare Dateien werden mit *execute* bezeichnet und *SCQL-files* beschreiben Datenbankdateien.

Zugriffssteuerung: Bei EF's wird der Zugriff über den Dateiheader gesteuert, während dier Zugriff auf DF's meist über das Betriebssystem gesteuert wird (jede Anwendung darf nur sein DF bearbeiten).

Ablaufsteuerung: Die Zustandsautomaten werden durch Zustandsgraphen (oder -tabellen) abgebildet, wobei *atomare Abläufe* in der Regel über Pufferspeicher realisiert werden, was allerdings teuer und langsam ist und deshalb nur für bestimmte EF's genutzt wird. *Nachladbarer Programmcode* besteht entweder aus Assembler Code oder wird interpretiert (→JavaCard). Da Chipkarten in der Regel keine MMU besitzen, stellt dieses Feature ein großes Sicherheitsproblem dar!

Ziel der Einführung der JavaCard war es, eine Plattform zu schaffen, mit der es möglich ist, einmal geschriebenen Programmcode auf allen Chipkarten lauffähig zu machen. Dazu wird der Source-Code in Byte-Code übersetzt und dann von der virtuellen Maschine auf der Karte ausgeführt. Diese virtuelle Maschine ist chipkarten-spezifisch und muß auf die jeweilige Karte angepasst werden. Die Einführung einer Zwischenschicht bei der Ausführung sorgt zwar für mehr Sicherheit, da Speicherzugriffe der Kontrolle der VM unterliegen, bringt jedoch auch für einen großen Einbruch in der Geschwindigkeit und stand der Verbreitung dieses Konzeptes bisher im Weg.

answer to reset - ATR

Reset der Karte, Protokoll-Auswahl

Quasi als Antwort auf den "Start" der Karte (Anlegen der Spannung und des Taktes und Senden des Reset-Signals), sendet die Chipkarte einen standardisierten Datenstring, der spezifische Informationen über den Takt der Karte und die Datenübertragung zwischen Karte und Host enthält. Auf Basis dieser Informationen muß dann mit Hilfe der *protocol type selection (PTS)* ein Kommunikations-Protokoll mit der Chipkarte ausgehandelt werden, indem bestimmte Parameter gesetzt werden. Als Übertragungsprotokolle stehen bei Prozessorchips die Standards *T=0, ... ,T=15* zur Verfügung, die sich z.B. durch Halb-/Vollduplexbetrieb oder Byte- und Blockorientiertheit unterscheiden.

application protocol data unit - APDU

Datenaustausch, Authentisierung möglich

Ein APDU bezeichnet genormte Dateneinheiten, die für den Datenaustausch zwischen Chipkarte und Host verwendet werden. Es wird unterschieden zwischen Kommando-APDUs und Antwort-APDUs. Da APDUs nur auf der Anwendungsschicht arbeiten, sind sie unabhängig vom Übertragungsprotokoll.

Um die Übertragung der APDUs vor Manipulationen zu schützen, kann man mit dem *authentic Verfahren* jedes Kommando mit einer Prüfsumme versehen, die dann an das Kommando angehängt wird. Dazu müssen beide Kommunikationspartner über einen gemeinsamen Schlüssel verfügen, da meist DES für die Erstellung der Prüfsumme verwendet wird. Dieser wird meist dynamisch erzeugt und am Anfang der Sitzung verschlüsselt übertragen.