Projektarbeit Internettechnologien

Kryptographische Verfahren

zur Datenübertragung im Internet

Dozent: **Dr. Kuhn**

Bearbeiter: **Patrick Schmid, Martin Sommer, Elvis Corbo**

Lehrgang: **12. Info - NTA Isny**

Isny, den 22.09.2005

Inhaltsverzeichnis

1 Grundlagen

1.1 Wie werden Daten übertragen?

Das Internet ist der Zusammenschluss vieler verschiedener Rechner. Beim Anfordern einer Webseite, wird die Anfrage daher über zahlreiche Server weitergereicht, bis sie an dem Computer ankommt, den Sie ansprechen wollen. Wenn man eine E-Mail schreibt, wird diese ebenfalls von Rechner zu Rechner gereicht, bis sie in dem Postfach des Servers landet, für den sie bestimmt ist.

Um zu verhindern, dass eine Übertragung abgehört oder sogar manipuliert wird, wurden für das Internet spezielle Protokolle entwickelt.

Zur Übertragung von Webseiten kommt zum Beispiel das SSL-Protokoll zur Anwendung. Dabei werden die Daten vom Browser zum Server verschlüsselt übertragen. Allerdings wird dieses Protokoll nicht immer genutzt, sondern meist nur in Fällen, in denen die zu übertragenden Daten besonders schützenswert erscheinen wie zum Beispiel Kontodaten oder Bestellungen in Internetshops. Man erkennt die Verwendung von SSL zum einen an der Adresszeile im Browser. Dort steht dann nicht ***http://*** sondern ***https://.*** Und zum anderen kann man es beim Internet Explorer oder dem Netscape Browser an dem kleinen Vorhängeschloss-Symbol in der Statuszeile erkennen.

Bild: Vorhängeschloss-Symbol in der Statuszeile.

Das SSL-Protokoll ermöglicht es, Daten vom Server sicher zu empfangen und Daten zum Server sicher zu senden, ohne dass jemand Kenntnis darüber erlangen kann.

Übertragung der Nachricht:
Eine verschlüsselte Nachricht muss in der Regel über mehrere Stationen übertragen werden. Heute handelt es sich dabei meist um einzelne Computersysteme, d.h. die verschlüsselte Nachricht wird über ein Computernetzwerk übertragen.
Man unterscheidet dabei zwei grundlegend unterschiedliche Übertragungsweisen: Leitungsverschlüsselung und Ende-zu-Ende-Verschlüsselung.

Bei der Leitungsverschlüsselung wird die Nachricht nur jeweils für den Nachbarcomputer verschlüsselt. Dieser entschlüsselt die Nachricht, verschlüsselt sie wiederum (mit einem möglicherweise anderen Verfahren) und schickt sie an seinen Nachbarn. Das geschieht so weiter bis zum Zielrechner. Der Vorteil dieses Verfahrens besteht darin, dass sich jeweils nur Nachbarrechner auf ein Verschlüsselungsverfahren und verwendete Schlüssel einigen müssen. Darüber hinaus kann diese Übertragungsweise auf einer sehr niedrigen Protokollebene (bereits in der Übertragungshardware) angesiedelt werden. Der Nachteil besteht darin, dass jeder einzelne Rechner auf dem Übertragungsweg vertrauenswürdig und sicher sein muss.

Bei der Ende-zu-Ende-Verschlüsselung wird die Nachricht vom Absender verschlüsselt und in dieser Form unverändert über mehrere Rechner hinweg zum Empfänger übertragen. Hier hat keiner der übertragenden Rechner Einsicht in den Klartext der Nachricht. Der Nachteil besteht allerdings darin, dass sich der Absender mit jedem möglichen Empfänger auf ein Verschlüsselungsverfahren und zugehörige Schlüssel einigen muss.

1.2 Was ist Verschlüsselung?

Verschlüsselung nennt man den Vorgang, bei dem ein Klartext mit Hilfe eines Verschlüsselungsverfahrens (Algorithmus) in einen Geheimtext umgewandelt wird. Als Parameter des Verschlüsselungsverfahrens werden ein oder mehrere Schlüssel verwendet.

1.2.1 Wieso wird verschlüsselt ?

„Wieso verschlüsseln? Ich habe nichts zu verbergen.“
Es geht gar nicht so sehr darum, ob man etwas zu verbergen hat. Aber manche Dinge möchte man lieber vertraulich behandeln. Wenn man zum Beispiel persönliche oder geschäftliche Nachrichten lieber im verschlossenen Umschlag statt als Postkarte verschickt, dann sollte es auch selbstverständlich sein, elektronische Post zu verschließen.

1.2.2 Kryptographie

Zufallszahlen sind kryptographische Basiselemente, über die am wenigsten gesprochen wird, die aber nicht weniger wichtig als die anderen sind. Fast jedes Computersicherheitssystem, das Kryptographie verwendet, braucht Zufallszahlen für Schlüssel, eindeutige Werte in Protokollen usw. und die Sicherheit dieser Systeme hängt oft von der tatsächlichen Zufälligkeit dieser Zufallszahlen ab. Wenn der Zufallszahlengenerator unsicher ist, versagt die ganze Kryptographie. [Zitat - John von Neumann: „Jeder, der arithmetische Methoden für die Produktion von Zufallsziffern in Erwägung zieht, befindet sich in selbstverständlich in einem Zustand der Sünde.“]

1.2.2.1 Verfahren der Kryptographie

In der Kryptographie werden zwei grundlegende Verfahren verwendet,
die Transposition und die Substitution.
Bei der Transposition werden die einzelnen Zeichen in eine andere Anordnung gebracht.
Bei der Substitution dagegen wird jedes Zeichen durch ein anderes Zeichen ersetzt.
Beides darf jedoch nicht wahllos geschehen, da der Empfänger diese Nachricht nicht in den Klartext zurück transformieren könnte. Die Regeln für die Transposition und Substitution werden in Algorithmen beschrieben.

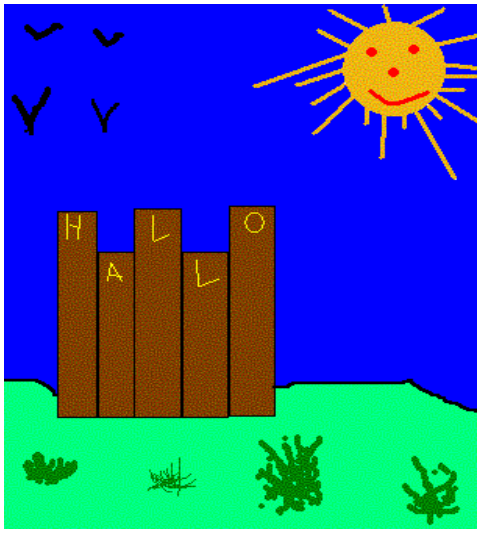
1.2.2.1.1 Transposition

Die Transposition beizeichnet ein Verschlüsselungsverfahren, bei dem die Zeichen des Klartextes umsortiert werden. Jedes Zeichen bleibt erhalten und ändert nur die Stelle, an der es auftritt.
Diese Form der Verschlüsselung findet Anwendung unter anderem beim Anagramm,

dem Gartenzaunprinzip und der Skytale.
Bei der Transposition bleibt der Klartext im Prinzip erhalten, es werden nur die einzelnen Zeichen in eine andere Anordnung gebracht, sie werden durcheinander gewürfelt.

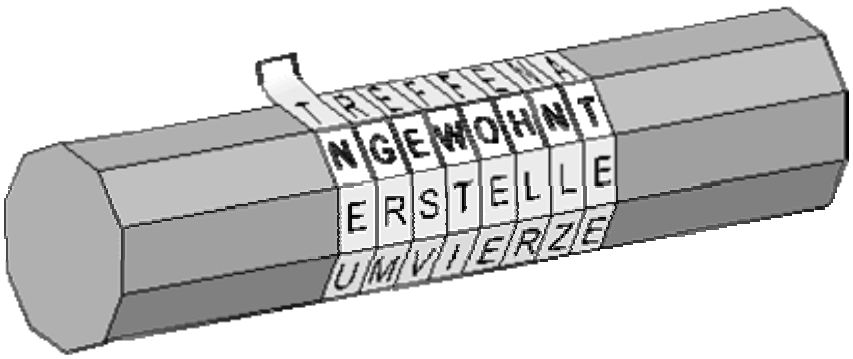
Anagramm
Ein Anagramm entsteht durch Umordnung der Zeichen.
Hinter dem Anagramm ***EDR*** können sich sinnvolle Worte wie ***DER*** oder ***RED*** verbergen

Gartenzaunprinzip



Das Gartenzaunprinzip ist in diesem Bild für das Wort ***HALLO*** dargestellt. Man schreibt den Text abwechselnd in die 1. bzw. 2. Zeile und fasst dann diese zu dem Codewort zusammen. Der Empfänger nutzt das Verfahren umgekehrt. Man kann natürlich auch noch mehr Zeilen benutzen, dann werden die Buchstaben noch mehr verwürfelt.
Die codierte Nachricht aus dem Bild wird zeilenweise gelesen und lautet: ***HLO AL***

Skytale
In dieser ersten Form der Transposition wurde ein Streifen Papier um einen Holzstab gewickelt und danach die Nachricht auf diesen Streifen geschrieben. Auf dem abgewickelten Streifen standen die Buchstaben der Nachricht, jedoch ist diese offenbar nicht einfach zu lesen. Wichtig war, dass der Empfänger und der Sender eine Skytale mit gleichem Durchmesser besaßen.



Wickelt man den Streifen ab, dann erhält man die Buchstabenfolge:
TNEURGRMEESVFWTIFOEEEHLRNNLZATEE

1.2.2.1.2 Substitution

Bei dem Verfahren der Substitution wird jeder Buchstabe durch einen anderen Buchstaben, eine Zahl oder ein Zeichen ersetzt. Eine eindeutige Abbildung des Klartextalphabetes auf das Geheimtextalphabet bezeichnet man als monoalphabetische Verschlüsselung. Von den einfachsten Anwendungen werden die Buchstabenpaarungen und die Caesar-Chiffre beschrieben.

Buchstabenpaarung
Auf der unteren Abbildung kann man eine mögliche Paarung von Buchstaben sehen. Wenn man hier das Wort ***HALLO*** verschlüsselt, erhält man ***UNYYB***. Man sucht den Buchstaben H und schaut welcher Buchstabe gegenüberliegt. Dieser wird an die Stelle des H gesetzt. Die Rücktransformation erfolgt analog.

Caesar – Chiffre
Bei der Caesar-Chiffre hat man ein Klartextalphabet, in welchem die zu sendende Nachricht geschrieben wird. Das Geheimtextalphabet enthält die Buchstaben in der Reihenfolge, in der sie ersetzt werden. Julius Caesar benutzte eine Verschiebung des Alphabets um drei Stellen. Es ist aber möglich Verschiebungen von 1 bis 25 Stellen vorzunehmen.

Klartextalphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Geheimtextalphabet	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Beispiel für Klartext: ***HALLO***

Dem Buchstaben ***H*** ordnet man den gegenüberliegenden Buchstaben ***K*** zu. Ebenso ersetzt man alle Klartextbuchstaben und erhält folgenden Geheimtext: ***KDOOR***

1.2.3 Steganografie

Stenographie oder Kurzschrift ist eine Schrift, die nach mehrjähriger Einübung so schnell schreibbar ist, dass mit ihr in normalem Tempo gesprochene Sprache mitgeschrieben werden kann. Der Begriff Stenographie leitet sich von den griechischen Wörtern stenos = eng und graphein = schreiben her.
Mit Kurzschriftsystemen können hohe Schreibgeschwindigkeiten erreicht werden, da der sprachliche Informationsgehalt mit grafischen und linguistischen Mitteln verzichtet wird.

2 Verschlüsselungsarten

Es wird grundsätzlich zwischen klassischen, symmetrischen, asymmetrischen und hybriden Verschlüsselungsarten unterschieden. **Klassische Verschlüsselungsarten** gab es schon unter Cäsar im Römischen Reich (die nach ihm benannte Caesar – Verschlüsselung). Ein weiteres Beispiel wäre das Vigenère – Verschlüsselungsverfahren. Den klassischen Verschlüsselungsarten liegt meist ein einfaches Prinzip zugrunde: die Verschiebung von Buchstaben/Zeichen, d.h. aus einem A wird ein C und aus einem B ein D (einfachste Form).

Da diese Verfahren mit Hilfe von Rechnern und Programmen wie **cryptool** relativ einfach geknackt werden können werden sie nur noch zu Demonstrationszwecken eingesetzt.

Um eine Nachricht, die mit einem Klassischen Verschlüsselungsverfahren codiert wurde, zu knacken wird eine Häufigkeit der im Chiffretext vorkommenden Buchstaben in Form einer Tabelle generiert und mit der relativen Häufigkeit der einzelnen Buchstaben eines Standardtextes verglichen. Der Rechner kann mit Hilfe der beiden Tabellen relativ einfach erkennen nach welchem Schema der Text verschlüsselt (verschoben) wurde.

Da solche Verfahren in der heutigen Zeit unbrauchbar geworden sind wurden neue Verfahren wie die symmetrische, asymmetrische und hybride Verschlüsselung entwickelt.

2.1 Symmetrisch

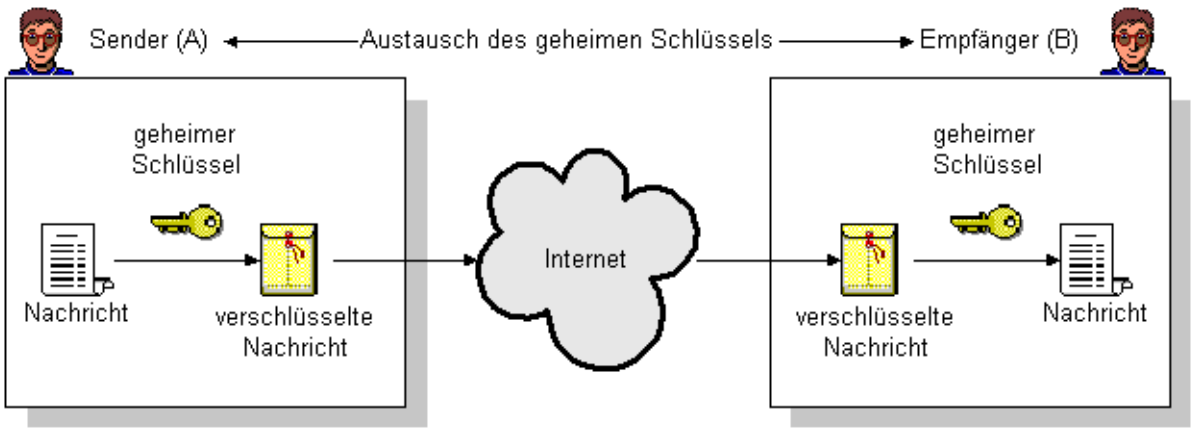
Bei der symmetrischen Verschlüsselung besitzen Sender und Empfänger exakt **den selben Schlüssel** zum codieren bzw. decodieren einer Nachricht. Symmetrische Verschlüsselungsverfahren lassen sich noch weiter unterteilen. Wird der Klartext Zeichen für Zeichen verschlüsselt spricht man von **Stromchiffren**. Teilt man den Klartext in Teile mit definierter Größe auf und codiert/decodiert somit mehr als ein Zeichen auf einmal spricht man von **Blockchiffren**.

Der Vorteil bei der symmetrischen Verschlüsselung ist die wesentlich höhere Sicherheit gegenüber klassischen Verschlüsselungsverfahren sowie bessere Performance gegenüber asymmetrischen Verfahren. Des Weiteren kann mit kürzeren Schlüsseln die selbe Sicherheit gewährleistet werden als mit vergleichsweise längeren asymmetrischen Schlüsseln.

Ein großer Nachteil der symmetrischen Verschlüsselung ist, dass jedes Paar von Personen, die separat von anderen Personen, geheime Nachrichten tauschen wollen jeweils einen eigenen Schlüssel benötigen. Für die geschützte Kommunikation von N Personen werden N*(N-1)/2 Schlüssel benötigt (4950 Schlüssel bei einer Gruppe von 100 Personen). Ein weiterer Nachteil ist, dass Schlüssel geheim bleiben müssen. Um absolute Sicherheit zu gewährleisten, ist es nicht möglich den Schlüssel seinem Kommunikationspartner über eine ungesicherte Leitung zu übermitteln.

Bekannte symmetrische Algorithmen sind:

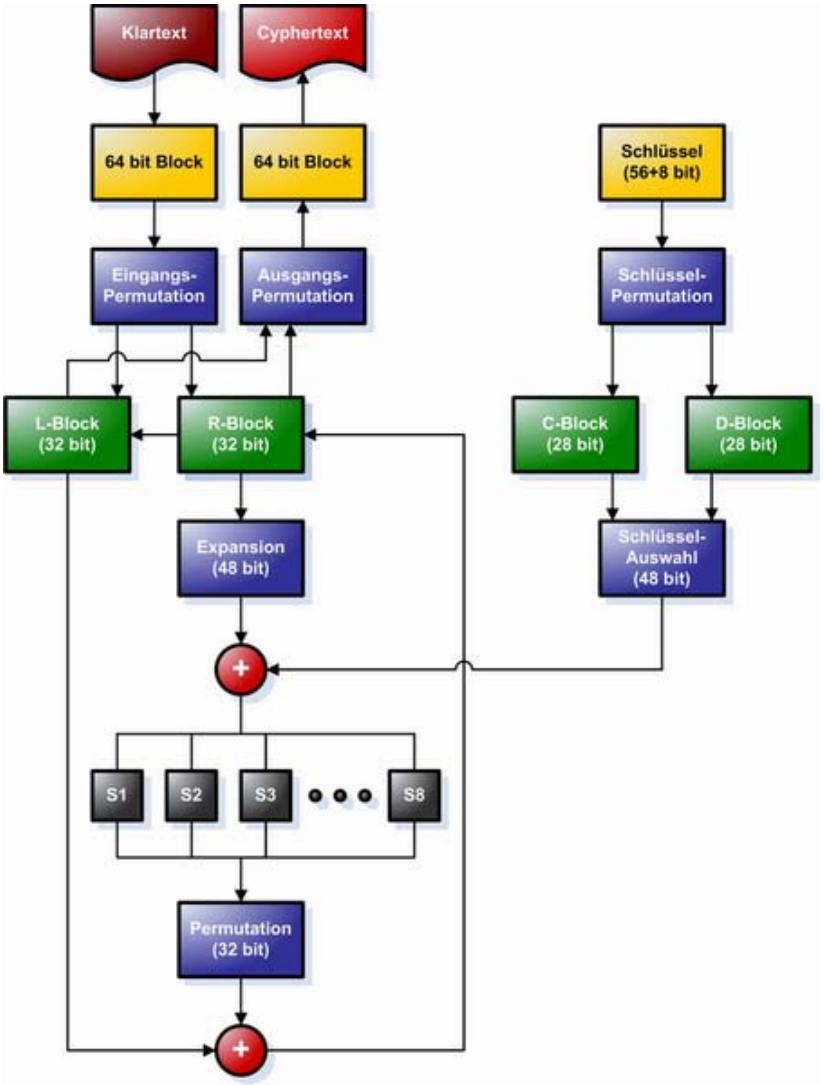
- DES
- 3DES
- AES
- IDEA
- Blowfish
- Twofish
- CAST
- Rijndael



2.1.1 DES / 3DES

DES (Data Encryption Standard) ist der bekannteste Algorithmus zur symmetrischen Verschlüsselung. DES wurde in den 70er Jahren erfunden und baut auf dem von IBM entwickelten System Lucifer auf. Es werden **Bitsubstitutions-, Bitpermutations-** und **Bitrekombinationsfunktionen** eingesetzt um einen Klartext auf Bit-Basis zu verschlüsseln.
64 Bit Klartext bzw. Chiffrenblock wird ein ebenfalls 64 Bit langer Schlüssel angewendet. Die effektive Länge des Schlüssels ist allerdings nur **56 Bit**, da die Bits n*8 (8,16,24,32,…64) als Paritätsbits benötigt werden. Mit 56 Bit lassen sich jedoch immer noch 70 Milliarden unterschiedliche Schlüssel realisieren.

Ablauf einer Verschlüsselung
Die Daten werden erst in 64 Bit große Blockchiffren aufgeteilt. Diese werden dann mittels **16 Iterationen** (Schritte) verschlüsselt, d.h. es werden auf die ursprünglichen Daten 16 Funktionen angewandt um den verschlüsselten Code zu erhalten, der ebenfalls 64 Bit lang ist. Abwechselnd werden Substitutionen und Transpositionen (Permutationen) nach dem **Schema von Feistel** durchgeführt. Beim Feistelverfahren wird die 64 Bit Blockchiffre in 2 gleich große Teile aufgeteilt und jeweils in beliebig vielen Durchgängen mit verschiedenen Schlüsseln verschlüsselt. Danach werden die 2 Teile wieder zusammengesetzt.
Bei der Decodierung werden die Schlüssel in gleich vielen Durchgängen in umgekehrter Reihenfolge verwendet.
Nachdem die Schlüssel wieder zusammengeführt wurden (Expansion) wird noch eine Substitutionsoperation durchgeführt. Diese wird auch als **S-Box** bezeichnet. Der DES – Algorithmus verwendet 8 solche S-Boxen. Sie ersetzen eine m-stellige Binärzahl durch eine n-stellige. S-Boxen werden meistens durch Tabellen implementiert. Sie haben die Funktion Klar- und Geheimtexte zu verwischen.



Beim DES – Algorithmus gibt es des weiteren **4 Betriebsmodi**:

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining Mode)
- OFB (Output Feedback Mode)
- CFB (Cipher Feedback Mode)

ECB (Electronic Code Book)

Der **ECB – Modus** ist der Einfachste. Die 64 Bit Klartextblöcke werden nacheinander und ohne Berücksichtigung bereits verwendeter Schlüssel codiert. Daraus resultieren allerdings große Sicherheitsmängel: Ganze Blöcke können ausgetauscht werden (z.B. den Block/die Blöcke mit Betrag/Empfänger einer Überweisung). Der Codierungsvorgang wird auch nicht „verwischt“. Es ist also einfacher einen Weg zur Decodierung zu finden.

CBC (Cipher Block Chaining Mode)

Bevor ein Klartextblock im **CBC – Modus** verschlüsselt wird, wird der bereits davor codierte Block zum Klartextblock dazu addiert. Da es beim ersten Klartextblock noch keinen Vorgänger gab wird hier ein geheimer Initialisierungsvektor (Timestamp oder voraussagbare, einfache steigende Zahl) addiert. Die Codierung geschieht also verkettet (-> chaining). Ein Nachteil dabei ist, dass nicht auf einen sondierten Block zugegriffen werden kann – es muss immer die ganze Datei decodiert werden. Vorteile sind die erhöhte Sicherheit (Angriffe werden erschwert da ein Block von allen vorhergehenden abhängt), die Verwischung des Klartextes und dass identische Klartexte unterschiedliche Chiffretexte als Ergebnis haben.

OFB (Output Feedback Mode)

Beim **OFB - Modus** wird der DES – Algorithmus nicht auf den Klartext angewandt. Stattdessen erzeugt der DES – Algorithmus einen Bitstrom der dann mit dem Klartext addiert wird (XOR – Operation ohne Übertrag). Der DES – Algorithmus wird von einem Schieberegister „gefüttert“, das mit einem bestimmten Initialisierungsvektor startet.

CFB (Cipher Feedback Mode)

Der CFB – Modus ist identisch mit dem OFB – Modus. In die Verkettung fließt jedoch an Stelle des codierten Bitstroms aus dem Schieberegister das Ergebnis der vorhergehenden Codierung ein.

Schwächen

Die relativ kleine Schlüssellänge von 54 Bit kann mittels Brute Force Angriffen geknackt werden. Es wird vermutet dass die NSA (bei der Entwicklung von DES beteiligt) absichtlich eine Schlüssellänge auswählte, die von ihren Rechnern in den 70er Jahren problemlos geknackt werden konnte. Durch die rasante Entwicklung ist es mit jedem herkömmlichen Computer möglich eine Brute Force Attacke mit Erfolg zu starten. Spezielle Rechnersysteme, die nur dem Zweck dienen DES – Codes zu

entschlüsseln können derzeit einen Schlüssel innerhalb von 22 Stunden 15 Minuten finden.
Aus diesem Grund erarbeiteten die DES – Entwickler den **3DES – Algorithmus**. Hierbei wird mit dem DES – Algorithmus ein Datenblock zuerst mit einem Schlüssel codiert, mit einem anderen decodiert und zu letzt wieder mit dem ersten Schlüssel chiffriert. Der Algorithmus wird dadurch deutlich sicherer jedoch auch wesentlich langsamer. Der Schlüsselraum vergrößert sich von 2^{56} auf 2^{112}.
Der oben beschriebene 3DES – Algorithmus wird Tuchman 3DES genannt es existieren aber auch noch andere Verfahren, die jedoch alle den DES – Algorithmus 3 mal anwenden (-> 3DES).

2.1.2 AES

Anfang 1997 rief das amerikanische Institut **National Institute of Standards and Technology (NIST)** zu einem öffentlichen Wettbewerb auf um einen geeigneten Nachfolger des DES bzw. des 3DES – Algorithmus zu finden. Der Sieger wurde dann als **Advanced Encryption Standard (AES)** veröffentlicht. Unter die fünf Besten kamen MARS, RC6, Rijndael, Serpent und Twofish. Am 2. Oktober 2002 ging der von Joan Daemon und Vincent Rijmen entwickelte **Rijndael – Algorithmus** als endgültiger Sieger hervor nachdem alle auf Schwachstellen überprüft worden waren. Der aus Belgien stammende Algorithmus überzeugte auch kritische Ausländische Kryptographen, die aufgrund der belgischen Herkunft überzeugt waren, dass die NSA keinen Einfluss auf den Code hatte.

AES wird als Blockchiffre klassifiziert. Der Code an sich überzeugt durch seine Sicherheit, Einfachheit und Performance. Eine Maschine die einen 128 Bit DES – Schlüssel in einer Sekunde knackt bräuchte für einen AES – Schlüssel gleicher Länge 149 Billionen Jahre. Er kann mit weniger als 500 Zeilen C-Quelltext implementiert werden.

<u>Ablauf der Verschlüsselung</u>
Der Klartext wird in Blöcke aufgeteilt, die eine Größe von 128 Bits besitzen. Die Größe war im ursprünglichen Algorithmus nicht vorgeschrieben sondern nur auf 32 Bit Schritte begrenzt. Unabhängig von dieser Größe wird ein Schlüssel mit einer Länge von **128, 192 oder 256 Bits** verwendet. Es wird eine zweidimensionale **Tabelle** generiert deren Zellengröße immer 8 Bit groß sind und mit einem Block gefüllt. Bei einer Größe von 128 Bit also 4 Spalten. Die Tabelle hat immer **4 Zeilen**. Auf jeden Block in der Tabelle werden verschiedene Operationen durchgeführt. Nun werden verschiedene **Teile des Schlüssels** nacheinander auf den Klartextblock angewandt. Die Anzahl der **Durchläufe (r)** ist abhängig von der **Schlüssellänge (k)** sowie von der **Blocklänge (b)**:

r	b = 128	b = 192	b = 256
k = 128	10	12	14
k = 192	12	12	14
k = 256	14	14	14

Zusammengefasst werden folgende Schritte durchlaufen:

- Schlüsselexpansion
- Vorrunde
  - **KeyAddition ()**
- Verschlüsselungsrunden (wiederhole solange runde<r)
  - **Substitution()**
  - **ShiftRow()**
  - **MixColumn()**
  - **KeyAddition()**
- Schlussrunde
  - **Substitution()**
  - **ShiftRow()**
  - **KeyAddition()**

Schlüsselexpansion
Hier werden die Teilschlüssel berechnet. Es werden r+1 Schlüssel benötigt. Er muss die gleiche Länge haben wie der Block den er verschlüsseln soll (**b*(r+1)**). Die Schlüssel werden ebenfalls in einer zweidimensionalen Tabelle gebildet. In der ersten Runde wird der Schlüssel, den der Benutzer gewählt hat, verwendet. Die Schlüssel der nächsten Runden werden berechnet. Die neuen Werte in den Zellen der ersten Spalte erhält man indem man das Byte in der letzten Spalte (entweder die 4., 6. oder 8. – je nach Schlüssellänge) und der 2. (danach: 3./4./1.) Zeile befindet hernimmt und mit der S-Box (Substitutionsbox die lediglich der Vermischung von Bytes dient, indem sie angibt welches Byte wie getauscht wird) verschlüsselt. Das Ergebnis wird dann mit dem um eine Schlüssellänge zurückliegende Byte XOR verknüpft. Anschließend wird **nur das erste Feld** des neuen Schlüssels mit einer Konstanten (aus einer so genannten rcon Tabelle), die für jeden neuen Schlüssel ändert, XOR verknüpft. Die weiteren Spalten werden aus den entsprechenden Spalten des vorherigen Schlüssels und der vorgehenden Spalte des entsprechenden Schlüssels mit XOR berechnet.

Vorrunde
KeyAddition()
In einer Vorrunde wird eine XOR-Verknüpfung mit dem Klartextblock und dem Schlüssel durchgeführt. Dies erschwert die know-plaintext Attacken.

Verschlüsselungsrunden
Jede Runde besteht aus folgenden Operationen:

- Substitution()
Byteweise Verknüpfung mit S-Box

- ShiftRow()
Zeilen werden um bestimmte Anzahl Spalten linksverschoben. Überlauf wird rechts fortgesetzt.

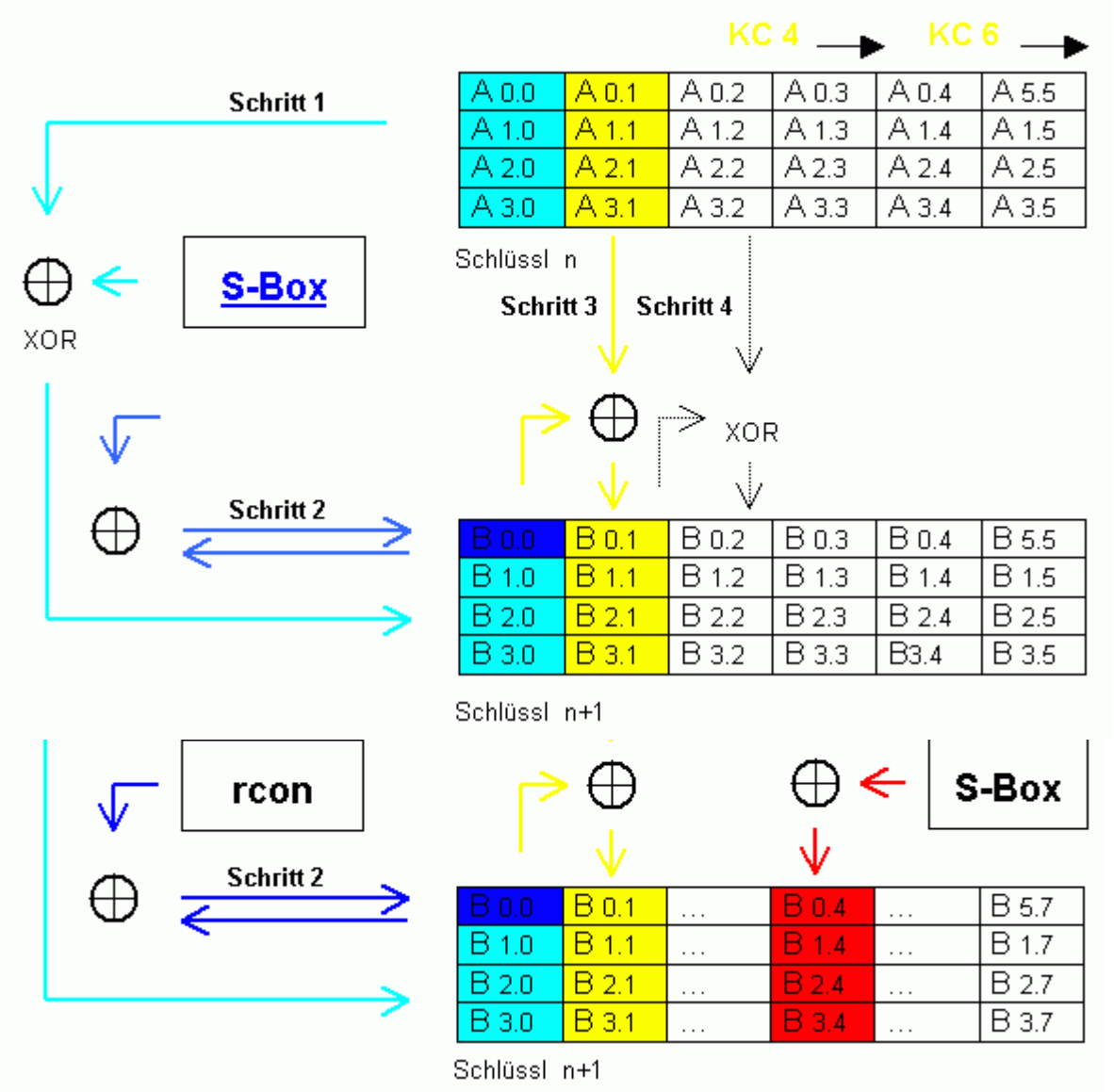
- MixColumn()

Die Spalten werden vermischt. Zunächst wir jede Zelle mit einer Konstanten multipliziert und dann XOR verknüpft. Die Wahl der Konstanten ist vordefiniert und hat mathematische Gründe:
Zeile 1: 2
Zeile 2: 3
Zeile 3: 1
Zeile 4: 1

- KeyAddition()
siehe oben

Schlussrunde
- Substitution()
- ShiftRow()
- KeyAddition()

Entschlüsselung
Alle Operationen werden in umgekehrter Reihenfolge aufgerufen. Das ist möglich weil es sich hauptsächlich um XOR Operationen handelt.
Es muss bei ShiftRow() nicht nach links sondern nach rechts verschoben werden.
Des Weiteren muss eine andere S-Box verwendet werden, die sich aus der ersten berechnen lässt.



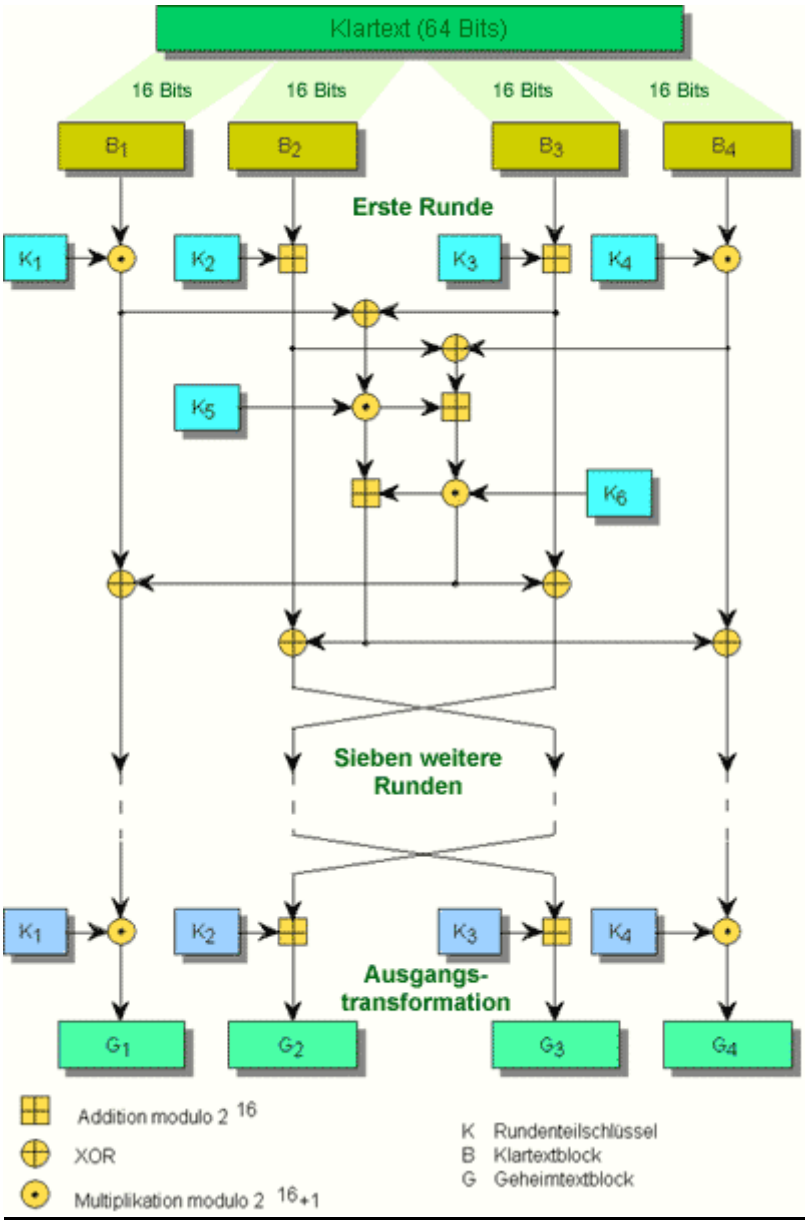
2.1.3 IDEA

Der **International Data Encryption Algorithm** (IDEA) wurde 1990 von der ETH Zürich und der Ascom Systec AG entwickelt. IDEA gehört wie DES und AES zu den Blockchiffren. Der **128 Bit Schlüssel** sowie der Klartext wird in Teile zerlegt. Der Klartext in **2 Teile mit jeweils 64 Bit**; der Schlüssel **in 8 Teile mit jeweils 16 Bit**. Die 2 Klartextteile werden später noch in 16 Bit Blöcke aufgeteilt.

Es werden 3 Operationen verwendet:

- XOR
- Addition modulo 2^{16}
- Multiplikation modulo $2^{16}+1$

Diese Operationen werden in 8 Runden immer Paarweise unter den 8 Teilen des Schlüssels durchgeführt:

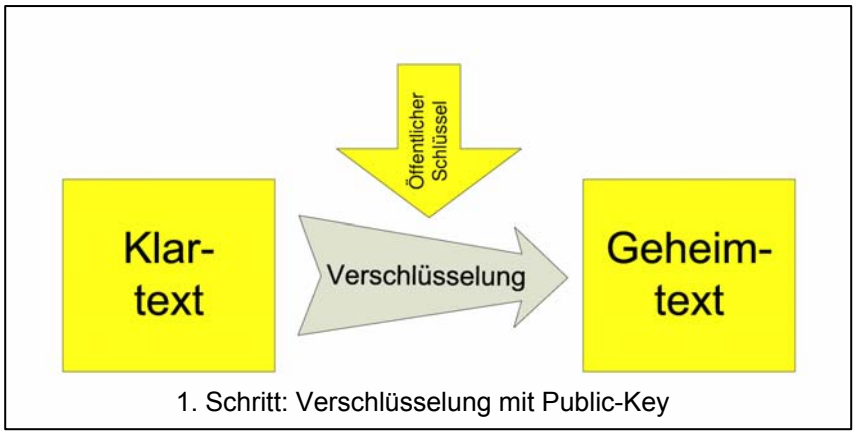


Vorteil:
Durch unterschiedliche mathematische Funktionen wird eine Attacke mittels einer **differentiellen Kryptoanalyse** erschwert.

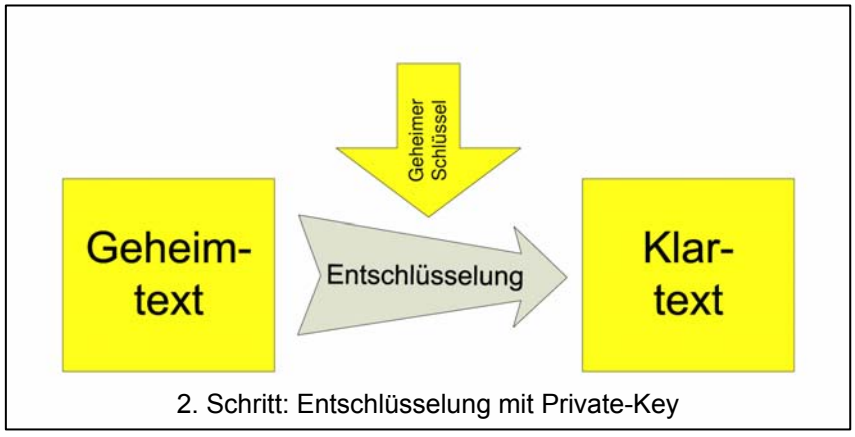
Nachteil:
Es existieren viele so genannte **schwache Schlüssel**. 2^{32} Schlüssel können mit einem gewählten Klartext nachgewiesen werden. Bei 2^{65} Schlüsseln kann man mit Hilfe von nur 20 gewählten Klartexten 72 Bit des Schlüssels ermitteln. Die übrigen 56 Bit kann man dann mit Brute Force ermitteln
Die Wahrscheinlichkeit, einen schwachen Schlüssel zu erwischen liegt allerdings bei eins zu 2^{63} (1 : 9 Trillionen).

2.2 Asymmetrisch

Das Problem der Schlüsselverteilung (s. symmetrische Verschlüsselung), dass keine Garantie gewährt, dass zwei Personen den gleichen Schlüssel besitzen, wird mit der asymmetrischen Verschlüsselung (oder auch Kryptographie mit öffentlichem Schlüssel bzw. Public-Key) umgangen.
Mit dieser Lösung kann man geheime Nachrichten an Leute schicken, die man nicht persönlich kennt und mit denen man sich nicht auf einen gemeinsamen Geheimschlüssel geeinigt hat.
Die Grundidee dabei ist, eine mathematische Funktion zu benutzen, die sich leicht in eine Richtung und schwierig in die andere berechnen lässt (diskreter Logarithmus). Die Faktorisierung von Ganzzahlen (Zerlegung in Faktoren) spielt hier eine wesentliche Rolle. Zwei Primzahlen kann man leicht miteinander multiplizieren und das Produkt ermitteln. Aber mit dem Produkt alleine ist es fast unmöglich, die Zahl zu faktorisieren und die zwei Primzahlen wiederzugewinnen.
Dieser Art von Mathematik wird meistens angewandt, um eine asymmetrische Verschlüsselung zu realisieren.
Statt eines einzigen Schlüssels, den Alice und Bob gemeinsam benutzen, gibt es hier zwei: einen für die Verschlüsselung (Encryption) und einen für die Entschlüsselung (Decryption). Die beiden Schlüssel sind unterschiedlich und es nicht möglich, einen aus dem anderen zu berechnen. Das heißt, wenn Eve den Verschlüsselungsschlüssel hat, kann sie nicht den Entschlüsselungsschlüssel herausfinden.



1. Schritt: Verschlüsselung mit Public-Key



2. Schritt: Entschlüsselung mit Private-Key

[Quelle: Wikipedia]

2.2.1 Diffie-Hellman

Der Algorithmus wurde von Martin Hellman gemeinsam mit Whitfield Diffie und Ralph Merkle an der Universität von Stanford (Kalifornien) entwickelt und 1976 veröffentlicht.
Er ist kein Verschlüsselungsverfahren, sondern beschreibt die Möglichkeit, Schlüssel sicher über unsichere Kanäle auszuhandeln. Er dient als Grundlage für das ElGamal-Verschlüsselungsverfahren.
Eine weitere Möglichkeit des Schlüsselmanagements ist das am MIT entwickelte Kerberos System mit sog. „Sitzungstickets“. Wir gehen hier aber nur auf ein Verfahren ein.

Vorbemerkung
a) Alice und Bob möchten miteinander verschlüsselt kommunizieren. Dazu möchten sie die Schlüssel für den Verschlüsselungsvorgang miteinander austauschen. Da keine gesicherte Verbindung zwischen den beiden besteht, kommt der Diffie-Hellmann Schlüsselaustausch zustande:

b) Definition Primitivwurzel:
Wenn p eine Primzahl ist, heißt die natürliche Zahl g (modulo p) Primitivwurzel, falls gilt:

- $0 \equiv (g^{p-1} - 1) \bmod p$
- Für alle natürlichen Zahlen $m < p-1$ ist $(g^m - 1) / p$ keine ganze Zahl

Oder anders ausgedrückt, wenn g die gesamte prime Restklassengruppe erzeugt.

Funktionsweise
Die Kommunikationspartner einigen sich zunächst auf eine Primzahl *p* und eine Primitivwurzel g, wobei:

$$g \bmod p \text{ mit } 2 \leq g \leq p-2$$

Diese Parameter müssen nicht geheim bleiben, können also insbesondere auch über ein unsicheres Medium übertragen werden.

Beide Kommunikationspartner erzeugen jeweils eine geheim zu haltende Zufallszahl a bzw. b aus der Menge {0, …, p - 2}. *a* und *b* werden nicht übertragen, bleiben also dem jeweiligen Kommunikationspartner, aber auch potenziellen Lauschern unbekannt.
Die Kommunikationspartner berechnen:

$$A = g^a \bmod p \text{ (Alice)}$$
$$B = g^b \bmod p \text{ (Bob)}$$

Nun werden A und B über das unsichere Medium an den Kommunikationspartner übertragen. (A an Bob bzw. B an Alice)
Die Kommunikationspartner berechnen nun:

$$\text{Alice berechnet } K = B^a \bmod p$$
$$\text{Bob berechnet } K' = A^b \bmod p$$
$$B^a \bmod p = (g^b \bmod p)^a \bmod p = (g^a \bmod p)^b \bmod p = A^b \bmod p = K$$

Es gilt: K = K'

Begründung:

$K = B^a \bmod p = (g^b \bmod p)^a \bmod p = (g^a \bmod p)^b \bmod p = A^b \bmod p = K'$

Das Ergebnis K ist für beide Partner gleich und kann als Schlüssel für die weitere Kommunikation verwendet werden

Beispiel
Kommunikationspartner sind wie immer Alice und Bob. Im Beispiel werden zur Vereinfachung kleine Zahlen genommen, in der Praxis sind wesentlich größere Primzahlen im Einsatz.

1. Alice und Bob einigen sich auf $p = 13$ und $g = 2$
2. Alice wählt $a = 5$. Bob wählt $b = 7$
3. Alice berechnet $A = 2^5 \bmod 13 = 6$.
   Bob berechnet $B = 2^7 \bmod 13 = 11$.
   Beide senden das Ergebnis an die jeweilige Gegenstelle.
4. Alice berechnet $K = 11^5 \bmod 13$, Bob berechnet $K' = 6^7 \bmod 13$. Beide erhalten das gleiche Ergebnis K = 7

2.2.2 DSA/DSS

Der Digital Signature Algorithm (DSA) wurde vom National Institute of Standards and Technology (NIST) im Rahmen des Digital Signature Standard (DSS) veröffentlicht, der 1994 veröffentlicht wurde. DSS wurde vom NIST in Zusammenarbeit mit der NSA ausgewählt, um einen elektronischen Unterschriftenstandard für die US-Behörden zu erhalten.
Bei DSA Verfahren ist die Generierung von Signaturen schneller als die Verifizierung, bei RSA ist es umgekehrt. DSA ist nicht wirklich sehr verbreitet, weil viele Nachrichten nur einmal signiert und öfter verifiziert werden.
Eingesetzt wird der Algorithmus bei SSL oder SSH.

Vorbemerkung
a) Definitionen:
Text (m), Private-Key (S), Public-Key (P), Hash-Verfahren SHA-1 (SHA)

b) Die Integer p, q und g können öffentlich sein. Private und Public-Key eines Users sind S und P. Parameter S und k werden nur zur Signatur Erzeugung benötigt und sollten geheim gehalten werden. Parameter k sollte für jede Signatur neu erzeugt werden.

Funktionsweise
Schlüsselpaar-Erzeugung:

Alice führt die folgenden Schritte durch:

Wähle Primzahlen p und q, so dass

- $2^{159} < q < 2^{160}$ ist;
- $2^{L-1} < p < 2^{L}$ für ein bestimmtes L, wobei L ein Vielfaches von 512 ist
- q teilt p-1

Berechne g für Signaturparameter:

$g = h^{(p-1)/q} \bmod p$, wobei h ein Integer ist mit $1 < h < p-1$, so dass $h^{(p-1)/q} \bmod p > 1$

Wähle Private Key:

S, wobei eine Zufallszahl (Integer) mit $0 < S < q$

Wähle Zufallszahl:

k, wobei eine Zufallszahl (Integer) mit $0 < k < q$

Berechne Public-Key mit Hilfe des Private-Keys:

$P = g^{S} * \bmod p$ (Berechnung von S:Diskreter Logarithmus Problem)

Signatur-Erzeugung:
Die Daten ergeben mittels eines Hash-Algorithmus (SHA-1 oder RIPEMD-160) eine Prüfsumme (Message Digest). Aus diesem Hash-Wert wird dann mit dem Private-Key die digitale Signatur erzeugt.
Digitale Signatur = $D_S(SHA(m))$

Alice führt die folgenden Schritte durch:

Berechnen der Modulo-Multiplikation mit Parametern p und q und einmaligem Signaturwert k:

$x = (g^{k} \bmod p) \bmod q$

Berechne y mit Berücksichtigung des Hash-Wertes des Textes und dem Private-Key:

$y = (k^{-1} (SHA(m) + S * x)) \bmod q$

=> Signatur von m ist (x, y)

Signatur-Prüfung:
Die Signatur-Prüfung berechnet aus der erhaltenen Nachricht den eindeutigen Hash-Wert. Mit diesem Hash-Wert und dem zugehörigen Public-Key berechnet DSS wieder die Echtheit der Signatur.
$Prüf_P$(*Digitale Signatur*) = SHA(m)
Liefert der Signatur-Prüf-Prozess ein positives Ergebnis, passen Signatur und Absender zusammen. Andernfalls stimmt die Signatur nicht mit dem Absender überein. Die Absender/Signatur wurde möglicherweise verändert.

Bob führt die folgenden Schritte durch:

Er benötigt zuerst den Public-Key P von Alice, um die Signatur zu überprüfen.

Dann berechnen von Verifikationswert mit Hilfe eines Signaturparameters:

$$w = y^{-1} \bmod q$$

sowie zwei Parameter, die für die Berechnung von v benötigt werden:

$$u1 = (SHA(m) * w) \bmod q$$
$$u2 = (x * w) \bmod q$$

und noch letztendlich, um die Gültigkeit von x berechnen zu können:

$$v = ((g)^{u1} * (P)^{u2}) \bmod p) \bmod q$$

Verifikation der Signatur (ausführlich):
Bob akzeptiert die Signatur wenn v = x ist.

Umformung von v:

$$v = ((g^{u1} * P^{u2}) \bmod p) \bmod q$$
$$= ((g^{(SHA(m) * w) \bmod q} * P^{(x * w) \bmod q}) \bmod p) \bmod q$$
$$= ((g^{SHA(m) * w} * P^{x * w}) \bmod p) \bmod q$$
$$= ((g^{SHA(m) * w} * g^{S * x * w}) \bmod p) \bmod q \qquad (\text{da } P = g^{S} * \bmod p)$$
$$= ((g^{w(SHA(m) + S * x)}) \bmod p) \bmod q$$

Auch ist:

$$y = (k^{-1} (SHA(m) + S * x)) \bmod q$$

und:

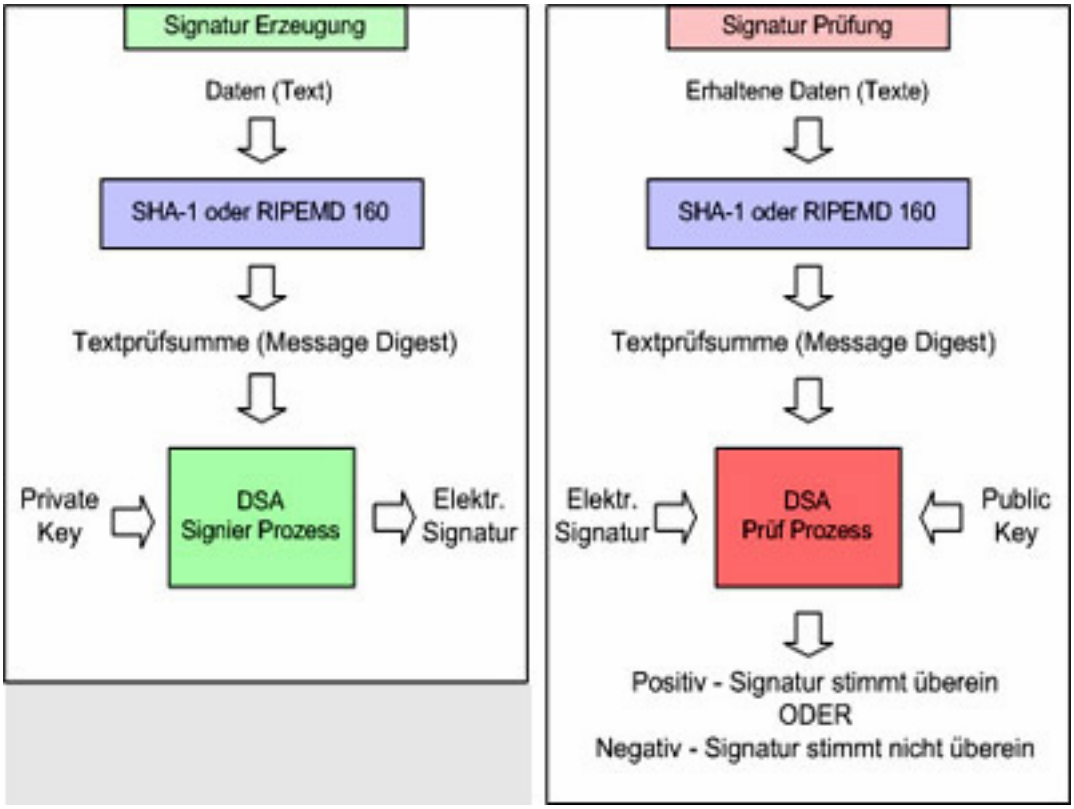
$$w = y^{-1} \bmod q$$
$$= (k(SHA(m) + Sx)^{-1}) \bmod q \qquad (\text{da } [? \bmod q] \bmod q = ? \bmod q)$$

$$(SHA(m) + Sx)w \bmod q = k \bmod q$$
$$w(SHA(m) + Sx) = k$$

Einsetzen der Zwischenergebnisse in v:

$$v = ((g^{k}) \bmod p) \bmod q$$
$$= x \qquad (\text{da } x = ((g^{k}) \bmod p) \bmod q)$$

=> Somit ist die Signatur gültig und wird von Bob akzeptiert.



[Erzeugungs- und Prüfprozess des DSA]

2.2.3 RSA

Der RSA-Algorithmus ist nach seinen drei Erfindern benannt: Ron Rivest, Adi Shamir und Leonard Adleman. Obwohl RSA eine der ersten Methoden für die Public-Key-Verschlüsselung war, ist er auch heute noch die am weitesten verbreitete Methode. Ein weiterer Vorteil ist, dass er nicht patentiert ist und somit weltweit ohne Einschränkungen genutzt werden kann.

Vorbemerkungen
Die nachfolgenden Bemerkungen bringen noch ein paar Definitionen mit sich:

a) Nachricht/Klartext (m), Private-Key (S), Public-Key (P), Chiffretext (c)

b) Das System funktioniert unter folgenden Bedingungen:
1. S(P(m)) = m für jede Nachricht m (grundlegende kryptographische Eigenschaft)
2. Alle Paare (S, P) sind verschieden (gewährleistet die Sicherheit)
3. Die Ableitung von S und P ist ebenso schwer wie das Entschlüsseln von m (gewährleistet durch diskreter Logarithmus)
4. Sowohl S als auch P lassen sich leicht berechnen (gewährleistet die praktische Anwendbarkeit)

c) Für die Public-Key-Kryptographie ist vor allem die Euler'sche Funktion φ wichtig. Sie gibt die Anzahl aller Zahlen zwischen 0 und n − 1, die zu n teilerfremd sind, an.

φ(5) = 4, da 1, 2, 3 und 4 teilerfremd zu 5 sind.
φ (12) = 7, da 1, 3, 5, 7, 9, 10 und 11 teilerfremd zu 12 sind.
φ (7) = 6, da 1, 2, 3, 4, 5 und 6 teilerfremd zu 7 sind.

Oder etwas mathematischer:
Eulersche phi-Funktion gibt Anzahl der Zahlen m aus Z_n an, für die ggT(m, n) = 1 gilt.

Es lässt sich also festhalten, dass für eine Primzahl p gilt: φ(p) = p − 1.
Ist aber n das Produkt zweier Primzahlen p und q, so gilt: φ(n) = (p − 1) * (q − 1).

d) Satz für Termumformung: $a^{1 \pmod{\varphi(n)}} = a \pmod{n}$

Funktionsweise
Damit Bob RSA-verschlüsselte Nachrichten von Alice erhalten kann, muss er sich als erstes einen öffentlichen Schlüssel generieren. Dazu muss er zuerst sich zwei (möglichst große, mind. zweistellig) Primzahlen zufällig auswählen und n bilden:

$$n = p * q$$

Als nächstes muss er eine natürliche, zu φ(n) = (p − 1) * (q − 1) teilerfremde Zahl P zufällig auswählen (Bedingung: P < n). Nun hat Bob seinen Public-Key:
die Zahlen P und n.
Dieser Public-Key kann gefahrlos weitergegeben werden. Auch wenn Eve ihn kennt, besteht keine Gefahr für Bob und Alice.
Nun muss er sich einen privaten Schlüssel mit folgender Formel generieren:

$$S = P^{-1} \pmod{\varphi(n)}$$

Jetzt erhält Alice den Chiffretext c, in den sie den Klartext m mit Bobs öffentlichen Schlüssel nach der Formel

$$c = m^P \pmod{n} \qquad \text{-> Verschlüsselung}$$

kombiniert. Diesen kann sie an Bob schicken.
Als letztes muss Bob nun den von Alice verschlüsselten Text noch entschlüsseln. Dazu verwendet er folgende Formel:

$$m = c^S \pmod{n} \qquad \text{-> Entschlüsselung}$$

Nun der Beweis:

$$\begin{aligned} C^S &= (m^P)^S \\ &= m^{P * (P\text{^}-1 \pmod{\varphi(n)})} \\ &= m^{1 \pmod{\varphi(n)}} \\ &= m \pmod{n} \end{aligned}$$

Beispiel

In unserem Beispiel will Alice die Zahl 20 an Bob übermitteln. Bob generiert nun als erstes die Primzahlen p = 5 und q = 23. Daraus folgt, dass

$$n = p * q = 115$$

Für die Zahl P nimmt er die 3, die teilerfremd zu φ(n) ist:

$$\varphi(n) = (p - 1) * (q - 1) = 4 * 22 = 88$$

Bobs öffentlicher Schlüssel besteht nun aus den Zahlen 115 und 3. Somit ist der Schlüssel P für die Verschlüsselung das Paar ganzer Zahlen (N, P) = (115, 3). Außerdem erstellt sich Bob noch seinen Private-Key mit:

$S = P^{-1} \pmod{\varphi(n)}$
$P * S = 1 \pmod{\varphi(n)}$
$3 * S = 1 \pmod{88}$
$3 * 59 = 1 \pmod{88}$

denn $177 = 1 \pmod{88}$
$177 = 1 + 2 * 88$

also ist der Schlüssel S für die Entschlüsselung das Paar (N, S) = (115, 59).

Nun verschlüsselt Alice die Nachricht (20) an Bob mit Hilfe des Public-Keys

$c = m^P \pmod{n}$
$c = 203 \pmod{115}$
$c = 65 \pmod{115}$

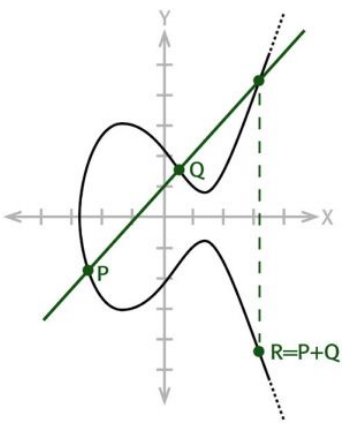
und überträgt ihn dann an Bob, der den verschlüsselten Text mit Hilfe seines geheimen Schlüssels entschlüsseln kann:

$m = c^S \pmod{n}$
$m = 65^{59} \pmod{115}$
$m = 20 \pmod{115}$

Er erhält wieder die ursprüngliche Nachricht (20).

2.2.4 ECC (Elliptic Curve Cryptography)

Elliptische Kurven, kurz EC (Elliptic Curves), sind keine Ellipsen, sondern werden so genannt, weil sie mit kubischen Gleichungen (z.B. $y^2 = x^3 + ax + b$) dargestellt werden. Sie basieren auf dem diskreten Logarithmus Problem.



Bereits im Jahre 1985 wurden diese von Victor Miller und Neal Koblitz unabhängig voneinander zur Konstruktion von asymmetrischen Kryptoalgorithmen vorgeschlagen. Die Vorteile von ECCs, die zukünftig für

Chipkartenrealisierungen bestimmt von großem Interesse sein werden, liegen einerseits in der weit geringeren notwendigen Rechenleistung (es wird kein eigener Krypto-Coprozessor benötigt) und andererseits in den kürzeren Schlüssellängen. Ein 160Bit Schlüssel eines ECCs entspricht in etwa dem eines 1024Bit RSA-Schlüssel, ein 320Bit Schlüssel bereits einem vergleichbaren 5120Bit RSA-Schlüssel. Daraus resultiert auch ein geringerer EEPROM Speicherbedarf für die Schlüssel. Der Algorithmus ist in der Praxis aber noch nicht ausreichend getestet; ob er sich bewährt, wird sich erst noch zeigen.
Ein Unterschied zu anderen asymmetrischen Verfahren liegt in der deutlich zunehmenden Komplexität der zugrundeliegenden Mathematik. Sind beim RSA (bzw. DSA) Algorithmus elementare Kenntnisse aus dem Bereich modularer Arithmetik für die Implementierung ausreichend, benötigt man im Falle der Kryptographie auf Basis elliptischer Kurven deutlich mehr Know-how bei der Umsetzung, insbesondere für die Bestimmung geeigneter Systemparameter.
Die Schwierigkeit für Angreifer liegt darin, dass passende k (Integer) zu finden, wenn man von gegebenen Punkten P und Q (s. Schaubild) ausgeht und die die Gleichung Q = kP aufstellen. Dabei muss die inverse Operation einer Punkt-Multiplikation berechnet werden. Auf den genaueren Algorithmus wird hier nicht weiter eingegangen.

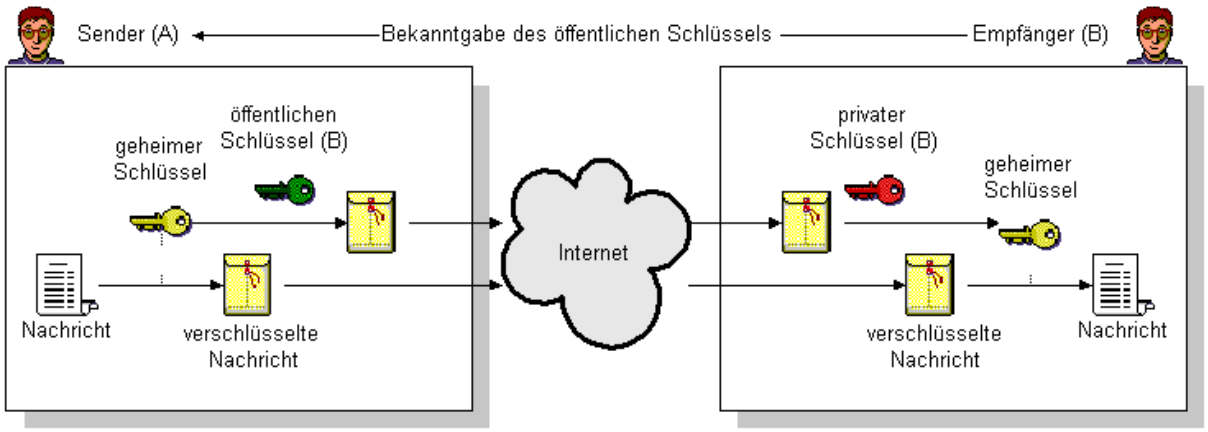
Der entscheidende Vorteil des ECC Verfahrens besteht darin, dass die bislang bekannten schnellen Algorithmen zur Lösung des Diskreten Logarithmus-Problems in endlichen Körpern (wie es z.B. dem DSA zugrunde liegt) in diesem Fall nicht anwendbar sind. Bei EC sind nur allgemeine Verfahren anwendbar, d.h. man kommt mit deutlich geringeren Schlüssel- und Parameterlängen aus, ohne Abstriche hinsichtlich der Sicherheit in Kauf nehmen zu müssen. Interessant ist das, wo Speicher- oder Rechenkapazität knapp sind, wie z.B. bei Smartcards und anderen Small Devices.
Diese Vorzüge führten zur Einbeziehung dieser Klasse von Verfahren in alle wesentlichen gegenwärtigen Standardisierungsbemühungen. Entsprechende Algorithmen fanden Berücksichtigung in den Entwürfen von IEEE, ANSI, ISO/IEC, DIN sowie dem deutschen Signaturgesetz.

2.3 Hybrid

Hybride Verfahren
Der Nachteil der asymmetrischen Verschlüsselung ist der hohe Rechenaufwand -> langsamer als symmetrische Verschlüsselung. Deswegen wird oftmals eine Kombination aus symmetrischer und asymmetrischer Verschlüsselung genutzt (s. 4.1 PGP). Dabei wird eine Nachricht durch den Empfänger zunächst mit einem speziellen geheimen Schlüssel (Session Key) symmetrisch verschlüsselt. Anschließend wird dieser Schlüssel mit dem öffentlichen Schlüssel des Empfängers asymmetrisch verschlüsselt und übertragen. Der Empfänger kann nun asymmetrisch mit seinem privaten Schlüssel den Session Key und somit die eigentliche Nachricht symmetrisch entschlüsseln. Da nur der asymmetrischen Verschlüsselung relativ gering.

Beispiele für den Einsatz von hybriden Verfahren:

- Secure Electronic Transaction (SET)
  - Standard für sichere Kreditkartenzahlung über das Internet
- Pretty Good Privacy (s. 4.1 PGP)
- …



[Vorgehenswseise beim hybriden Verschlüsselungsvorgang]

3 Protokolle zur sicheren Übertragung

Die sechs Basisverfahren: Symmetrische Verschlüsselung, Message Authentication Code, Public-Key Verschlüsselung, Einweg-Hash Funktionen, digitale Signaturen und Zufallsgenerator bilden den Kern jedes kryptographischen Protokolls. Nachfolgend einige der wichtigsten und gebräuchlichsten Kryptoprotokolle.

3.1 SSL (Secure Socket Layer)

SSL ist eines der am weitesten verbreiteten Protokolle, das sichere Datenübertragung im Internet gewährleistet. Die Daten werden sicher und unverfälscht vom Sender an den richtigen Empfänger übermittelt.
SSL wird neuerdings seit SSL Version 3.0 **TLS (Transport Layer Security)** genannt. Das erste Release von SSL wurde im November 1993 in den damals weit verbreiteten Browser Mosaic integriert. Im August 1994 implementierte Netscape Communications Version 1.0 in ihrem Browser. Mit dem neuen Netscape – Browser fünf Monate später wurde SSL – Version 2.0 veröffentlicht. 1995 entwickelte Microsoft ein Äquivalent namens PCT (Private Communication Technology) für den Internet Explorer. PCT hatte einige Vorteile gegenüber SSL 2.0. Diese wurden aber mit der neuen Version 3.0 wieder irrelevant. 1999 wurde SSL 3.0 mit geringfügigen Veränderungen von der IETF (Internet Engineering Task Force) unter dem Namen TLS (Transport Layer Security) 1.0 standardisiert. Um Konflikte zu vermeiden wurde im Header von TLS 1.0 Version SSL 3.1 angegeben. TLS wurde laufend durch RFCs (Request for Comments) erweitert:

RFC 2721 – Addition of Kerberos Cipher Suites
RFC 2817 – Upgrading to TLS Within HTTP/1.1
TLS über bestehende TCP – Verbindung (Ports 80 bzw. 443)
RFC 2818 – HTTP over TLS
Trennung von sicherem und unsicherem Verkehr durch selbst definierten TCP Port
RFC 3268 – Advanced Encryption Standard (AES) Ciphersuites
Die symmetrischen Verschlüsselungsalgorithmen DES, 3DES, AES, IDEA, RC2 und RC4 wurden implementiert

Funktion
SSL arbeitet in den Schichten zwischen Transport- und Application Layer des OSI – Modells. Dadurch kann SSL transparent fungieren und somit einfach „hinzugeschalten“ werden. Der Einsatz von SSL mit Protokollen ohne Sicherheitsvorkehrungen wird somit ermöglicht. SSL besteht aus zwei Schichten: **SSL Handshake Protocol** und **SSL Record Protocol**.

Das SSL Handshake Protocol kommt zum Einsatz **bevor** die ersten Nutzdaten gesendet werden.

Der SSL Record Protocol Layer setzt direkt auf der Transportschicht des OSI – Modells auf. Sie bietet zwei Operationen an, die unabhängig von einander genutzt werden können.

Eine SSL benötigt den Einsatz von Zertifikaten da die Verbindung mit zwei unterschiedlichen Verschlüsselungstechniken arbeitet. Zertifikate sind elektronische Ausweise mit denen die Identität eines Computers oder einer Person bestätigt werden kann. Für die sichere Transaktion im Internet werden Trust Center eingesetzt.

Für eine sichere SSL - Verbindung muss ein Trust Center Folgendes bereit stellen:

- einen öffentlichen Schlüssel (public key)
  - ist für alle Nutzern im Internet zugänglich
- einen privaten Schlüssel (private key)
  - ist geheim und nur dem Nutzer bekannt.
  - wird zum decodieren verschlüsselter Nachrichten verwendet
- ein Zertifikat
  - elektronischer Ausweis der darüber informiert zu welchem Computer/ welcher Person ein Schlüssel gehört

Zertifikate sind meist bereits im Browser integriert, können aber auch später installiert werden. Jeder Server im Internet erhält vom Trust Center ebenfalls ein Zertifikat welches den Namen des Servers und dessen öffentlichen Schlüssel beinhaltet. Das Zertifikat des Servers wird mit dem privaten Schlüssel des Trust Centers verifiziert. Das macht ein Zertifikat fälschungssicher.

Aufbau einer Verbindung

Um eine Verbindung aufzubauen müssen zunächst vom Client sowie vom Server Informationen ausgetauscht werden. Der Client teilt dem Server seinen Kommunikationswunsch mit indem er ihm eine Nachricht schickt. Der Server schickt dieselbe Nachricht zurück und sendet sein Zertifikat. Der Client überprüft die Identität des Servers indem er das Zertifikat mit dem öffentlichen Schlüssel des Trust Centers überprüft, das im Zertifikat seines Browsers integriert ist. Tritt kein Fehler auf kann man sicher sein, dass man mit dem richtigen Server kommuniziert.
Der Client erzeugt nun einen so genannten **Pre - Master Secret**, der mit Hilfe des öffentlichen Schlüssel des Servers codiert wird. Dieser Pre - Master Key wird an den Server geschickt, der ihn mit seinem private Key entschlüsseln kann. Er erstellt daraus das **Master Secret**. Aus dem Master Secret wird von beiden Seiten ein **Session Key** erzeugt.
Nun steht die Verbindung und es können Nutzdaten gesendet werden.

Übertragung der Daten

Die Nachricht wird mit dem Session Key verschlüsselt und an den Kommunikationspartner versendet. Der Empfänger entschlüsselt die Nachricht wiederum mit demselben Session Key.

Einsatz:

Das SSL Verfahren wird vor allem bei **HTTPS**. Auf den meisten Webservern ist SSL 2.0 sowie SSL 3.0 über TLS vorhanden bevorzugt wird aber SSL 3.0/TLS. Bevorzugt werden auch die Verschlüsselungsverfahren AES und RSA.
OpenSSL ist eine Open Source Variante des herkömmlichen SSL. Es unterscheidet sich durch weiterentwickelte Methoden zur Zertifikatverwaltung und unterschiedlichen Verschlüsselungsfunktionen.

3.2 SSH (secure shell)

Die 1995 von Tatu Ylönen aus Finnland erfundene SSH ist eine Kombination aus **Netzwerkprotokoll** und **Programm**. Mit Hilfe von SSH ist es möglich, sich auf einem anderen Computer im Internet anzumelden. Auf diesem entfernten Computer können dann Programme gestartet oder Daten getauscht werden. Die Benutzung von X11 (X-Window System von UNIX) ist ebenfalls möglich. Die Verbindung zwischen den 2 Rechnern wird **verschlüsselt**. Den gleichen Dienst ohne Verschlüsselung erbringen die Programme rsh, rlogin und telnet. Im Gegensatz zum Ursprungs SSH, das es aufgrund der guten Dokumentation Angeifern leicht machte, verzichtet das veröffentlichte SSH auf die Verschlüsselungsalgorithmen IDEA und DES.
IDEA kann nur in bestimmten Fällen kostenlos implementiert werden und DES genügte nicht den Sicherheitsanforderungen. Stattdessen wird der 3DES (Triple DES) Algorithmus verwendet. Beim Verbindungsaufbau gewährleistet RSA, dass mit dem richtigen Server kommuniziert wird.

Von der IANA (Internet Assigned Numbers Authority) wurde diesem Dienst die **TCP - Portnummer 22** zugeteilt. SSH fungiert in der Schicht direkt über TCP (Transport Layer). SSH wurde für UNIX – Systeme entwickelt; kann aber genauso unter Windows beispielsweise über ein Hilfsprogramm namens **PuTTY** genutzt werden, das eine Terminalsitzung emuliert.

In der neueren Version des Protokolls SSH2 können im Gegensatz zu SSH1 mehrere Verschlüsselungsalgorithmen verwendet werden. Sicherheitsprobleme die mit der Integritätsprüfung von SSH1 zusammenhingen wurden ebenfalls ausgemerzt. Die Version 1.x und 2.x sind nicht kompatibel !

OpenSSH und **OSSH** ermöglichen die kostenlose Nutzung von SSH. Sie enthalten Erweiterungen sowie Verbesserungen des Ursprungs – Codes, der in C geschrieben wurde. OpenSSH besitzt keine eigenen Verschlüsselungsalgorithmen sondern verwendet die, die in OpenSSL implementiert sind.

SSH – Kommandos:

ssh / slogin	- erzeugt Verbindung
scp	- sicheres Kopieren von Files
ssh-keygen	- erzeugt einen RSA - Schlüssel
ssh-agent	- verwaltet private RSA – Schlüssel -> automatisierte Authentifizierung über einen Agent (Hilfsprogramm)
ssh-add	- registriert neue RSA – Schlüssel beim Agent
make-ssh-known-hosts	- erstellt Liste mit öffentlichen Host – Keys einer Domäne

ssh [**-l** *login_name*] *hostname* | *user@hostname* [*command*]

ssh [**-afgknqstvxACNPTX1246**] [**-b** *bind_address*] [**-c** *cipher_spec*]
[**-e** *escape_char*] [**-i** *identity_file*] [**-l** *login_name*] [**-m** *mac_spec*] [**-o** *option*]
[**-p** *port*] [**-F** *configfile*] [**-L** *port*:*host*:*hostport*] [**-R** *port*:*host*:*hostport*]
[**-D** *port*] *hostname* | *user@hostname* [*command*]

Terminal - Kommandos
~. Verbindung beenden
~^Z ssh in Hintergrund legen
~# Auflistung der weitergeleiteten Verbindungen
~& ssh beim Ausloggen in den Hintergrund legen ohne X11 Beendigung abzuwarten
~? Liste von Escape - Sequenzen
~R Erzwingt Neuverschlüsselung der Verbindung

Ablauf der Kommunikation
- Über den Port 22 wird eine TCP-Verbindung vom Client zum Server aufgebaut.
- Protokoll – Versionsnummern werden getauscht – Abbruch bei Inkompatibilität
- Client und Server schalten auf paketbasiertes Binär-Protokoll um mit folgenden Informationen:
  - Paketlänge: max. 262144 Byte
  - Padding: 1 bis 8 Byte Zufallsdaten (Verhindert known-plaintext Attacken)
  - Pakettyp
  - Nutzdaten
  - Püfsumme (CRC)
- Server sendet seine beiden öffentlichen RSA – Schlüssel (Host- u. Server- Key) und unterstützte Verschlüsselungsalgorithmen
- Client verifiziert Host Key und generiert Session Key
- Client sendet dem Server den mit dem Host- und Server- Key verschlüsselten Session Key und den von ihm gewählten Verschlüsselungsalgorithmus an den Server. -> **verschlüsselte Verbindung besteht** !
- Client loggt sich auf Server ein
- Server stellt Arbeitsumgebung zur Verfügung (setzt Umgebungsvariablen)
- Austausch von Nutzdaten beginnt

Verfahren zur Authentifizierung der Kommunikationspartner

1. *Rhosts-Authentifizierung*

   - wie bei rlogin und rsh
   - basiert auf Einträgen in den Dateien `/etc/hosts.equiv` oder `/etc/shosts.equiv` bzw. `$HOME/.rhosts` oder `$HOME/.shosts`.
   - ist unsicher

2. *Rhosts-RSA-Authentifizierung*

   - Kombination von (1) und RSA bsierter Authentifikation
   - Öffentliche Schlüssel der Clients auf Server in den Dateien `$HOME/.ssh/known_hosts` und `/etc/ssh_known_hosts`
   - Client weißt nach dass er zugehörigen private Key kennt

3. *reine RSA-Authentifizierung*

   - Nutzer weißt die Kenntnis seines privaten Schlüssels über Challenge-Responsee-Verfahren nach
   - auf dem Server im File `$HOME/.ssh/authorized_keys`.
   - Das Schlüsselpaar des Nutzers ist in `$HOME/.ssh/identity` gespeichert.

4. *Paßwort-Authentifizierung*

   - Authentifizierung durch verschlüsselt übertragenes Nutzerpaßwort
   - Standard: UNIX – Passwort
   - Alternativ: AFS-/Kerberos- Passwörter oder Token-Codes von ScurID-Karten

3.3 SFTP

SFTP (**S**ecure **F**ile **T**ransfer **P**rotocol) ist ein Netzwerkprotokoll das eine sichere Datenübertragung auf TCP/IP Basis ermöglicht. Realisiert wird dies durch eine Verbindung von den Standardprotokollen FTP und SSH. Beim ungesicherten FTP erfolgt das Authentifizieren, Auflisten von Dateien und Wechseln von Ordnern unverschlüsselt über Port 21. Bei SFTP wird diese Verbindung **über SSH getunnelt**. Die eigentlichen Daten werden über einen zufällig gewählte Port unverschlüsselt übertragen.

Verschlüsselte Datenübertragungen ermöglichen SSH FTP und SCP (Secure Copy Protocol).

Syntax:

sftp [**-vC1**] [**-b** *batchfile*] [**-o** *ssh_option*] [**-s** *subsystem* | *sftp_server*]
[**-B** *buffer_size*] [**-F** *ssh_config*] [**-P** *sftp_server path*] [**-R** *num_requests*]
[**-S** *program*] *host*
sftp [[*user*@]*host*[:*file* [*file*]]]
sftp [[*user*@]*host*[:*dir*[*/*]]]

scp -C2 -P 222 -i ~/.ssh/id_rsa|dsa user@server.dyndns.org:[/pfad/]remote-Dateiname [/lokaler/pfad/]lokaler-Dateiname

Option –C aktiviert Kompression

3.4 IPSec: security architecture for IP

Das IPsec-Protokoll wurde 1998 entwickelt, um die Schwächen des Internetprotokolls (IP) zu beheben. Es stellt eine Sicherheitsarchitektur für die Kommunikation über IP-Netzwerke zur Verfügung. Das Protokoll soll Vertraulichkeit, Authentizität und Integrität gewährleisten. Daneben soll es vor so genannten Replay-Angriffen schützen, d.h. ein Angreifer kann nicht durch Abspielen eines vorher mitgeschnittenen Dialogs die Gegenstelle zu einer wiederholten Aktion verleiten.

IPsec entstand im Zuge der Entwicklung von IPv6 und ist in verschiedenen RFC´s (Request For Comments) spezifiziert:

RFC 2401 : Sicherheitsarchitektur für das Internetprotokoll (RFC 2401)
RFC 2402 : Authentication Header (RFC 2402)
RFC 2406 : Encapsulating Security Payload (RFC 2406)
RFC 2407 : IPsec Domain of Interpration (IPsec DoI, RFC 2407)
RFC 2408 : Internet Security Association and Key Management Protocol
RFC 2409 : Internet Key Exchange (IKE, RFC 2409)

Der RFC 2401 bildet das Hauptdokument zu IPsec. Von dort aus werden die oben genannten RFC´s referenziert.
Wesentliche Inhalte von IPsec sind das Authentication Header Protokoll (AH) und das Encapsulated Security Payload Protokoll (ESP) sowie das Protokoll zum Austausch der Schlüssel.

Im Gegensatz zu anderen Verschlüsselungsprotokollen wie etwa SSH arbeitet IPsec auf der Netzwerkschicht (Schicht 3) des OSI Referenzmodells.

Schlüsselverwaltung (IKE)
Vor dem eigentlichen Start einer verschlüsselten Verbindung muss man sich über die zu verwendenden Schlüssel und Algorithmen klar werden. Hierfür ist IKE gedacht. IPsec arbeitet mit verschiedenen symmetrischen wie asymmetrischen Schlüsseln. Schlüssel können manuell oder automatisch ausgehandelt werden.
Für die automatische Schlüsselverwaltung wurde das Internet Key Exchange (IKE) Protokoll entworfen. IKE ist in RFC 2409 spezifiziert und basiert auf dem Internet Security Association and Key Management Protokoll (ISAKMP, RFC 2408), der IPsec Domain of Interpretation.

IKE basiert auf UDP (User Datagram Protocol) und nutzt standardmäßig den Port 500.
Es arbeitet in zwei Phasen:

a) Aushandlung einer Security Association (SA) für ISAKMP mittels Aggressive Modus oder Main Modus.
b) Erzeugung einer Security Association für IPsec mittels Quick Modus.

Eine Security Association ist ein Vertrag zwischen den kommunizierenden Stellen. Dort wird festgelegt, welche Authentifizierungs- und Verschlüsselungsalgorithmen genutzt werden sollen.

Main Modus
Der Main Modus kann in der ersten Phase der Internet Key Exchange genutzt werden. Hierbei handelt der Initiator (derjenige, der die Verbindung aufnehmen will) und der Antwortende miteinander SA´s für ISAKMP aus. Diese Verhandlung geschieht in folgenden sechs Schritten:

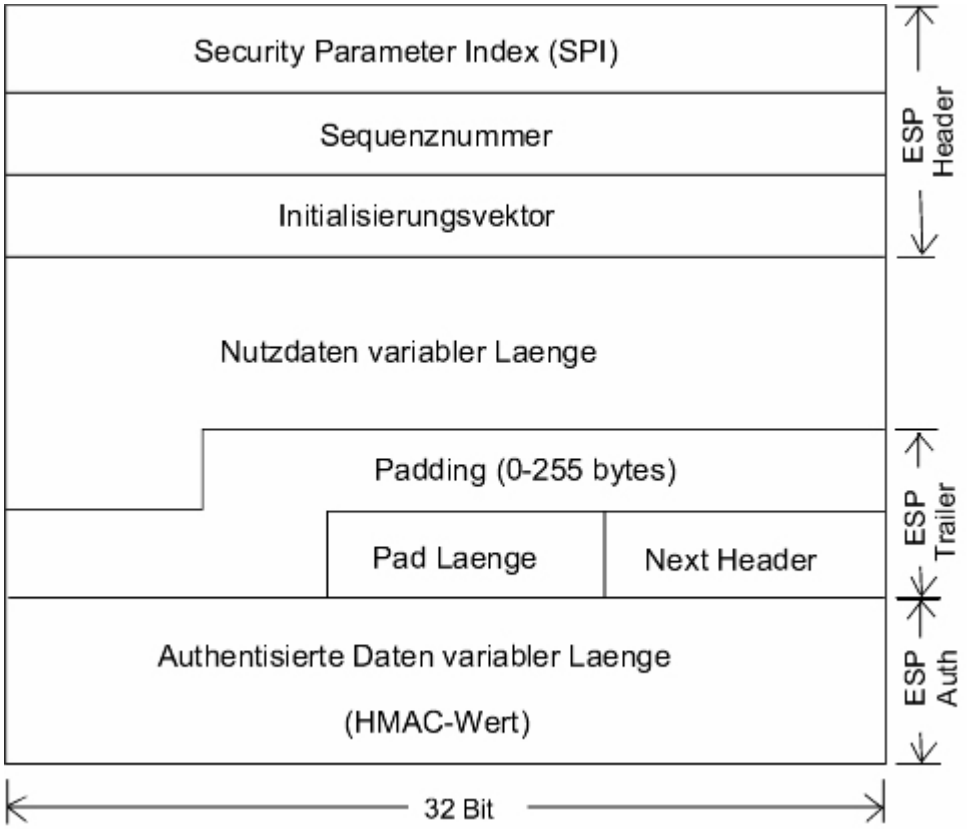
1. Initiator sendet einen oder mehrere Vorschläge mit Authentifizierungs- und Verschlüsselungsalgorithmen.
2. Antwortender wählt einen Vorschlag aus und bestätigt.
3. Initiator sendet öffentlichen Teil der Diffie-Hellmann-Schlüsselvereinbarung und einen zufälligen Wert.
4. Antwortender schickt ebenfalls öffentlichen Teil der Diffie-Hellmann-Schlüsselvereinbarung und einen zufälligen Wert.
5. Initiator berechnet Signatur und sendet diese mit seiner Identität an Antwortenden. Diese Daten werden mit einem symmetrischen Schlüssel verschlüsselt.
6. Antwortender schickt gleiche Daten von seiner Seite an den Initiator Aggressiven Modus.

Im Aggressiven Modus werden die obigen Schritte auf drei zusammengefasst. Hierbei fällt dann die Verschlüsselung des fünften Schrittes weg. Stattdessen werden die Werte im Klartext übertragen. Daher sollte man diesen Modus nach Möglichkeit nicht verwenden.

Quick Modus
Der Quick Modus wird in der zweiten Phase von IKE zur Anwendung gebracht. Die gesamte Kommunikation in dieser Phase erfolgt verschlüsselt. Wie in der ersten Phase wird zunächst ein Vorschlag (Proposal) gemacht. Dieser wird zusammen mit einem Hashwert und der Nonce übertragen. Später werden die Schlüssel neu berechnet und es gehen keinerlei Informationen aus den zuvor generierten SA´s ein. Dies stellt sicher, dass niemand von der zuvor generierten Schlüsseln auf die neuen schließen kann.

Authentication Header
Authentication Header (AH) soll die Integrität und Authentizität der übertragenen Pakete sicherstellen. Weiterhin existiert hier ein Schutz gegen Replay-Angriffe. Der Authentication Header versucht alle möglichen Felder eines IP-Datagramms zu schützen. Es werden lediglich Felder ausgeschlossen, die sich während des Laufes eines IP-Paketes ändern können.

Ein AH-Paket sieht folgendermaßen aus:



Bedeutung der Felder:

- SPI : identifiziert in Verbindung mit der IP-Adresse und dem Sicherheitsprotokoll die Sicherheitsassoziation
- Sequenznummer: ansteigende Nummer, die vom Absender gesetzt wird, soll Schutz vor Replay-Angriff bieten
- Nutzdaten: enthält die Datenpakete
- Füllung: wird für eingesetzte Blockchiffre genutzt, um Daten bis zur vollen Größe des Blocks aufzufüllen
- Länge Füllung: enthält Anzahl der eingefügten Bits für Padding
- Nächster Header: identifiziert das Protokoll, der im Paket übertragenen Daten

Authentizitätsdaten: enthält den Wert des Integritätstests

3.5 S/MIME (Secure/Multipurpose Internet Mail Extensions)

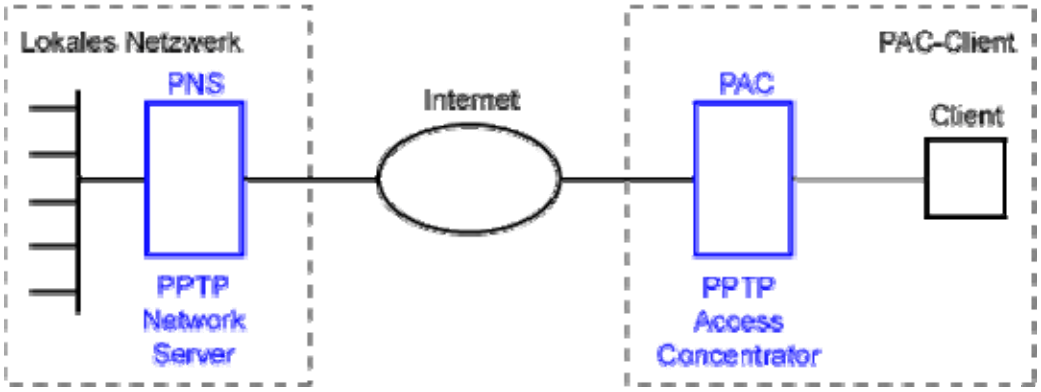
Als bislang sicherste Methode für E-Mails gilt die Verschlüsselung mit PGP**.**
In der Praxis kommt PGP aber eher selten zum Einsatz, denn dieser Verschlüsselungsmethode haftet immer noch der Ruf einer wenig benutzerfreundlichen Technik an, auch wenn dem längst nicht mehr so ist.
S/MIME ist ein relativ neues Protokoll und soll für den sicheren Austausch von E-Mails sorgen. Es wurde bereits in verfügbare Anwendungen wie Microsoft Outlook, Outlook-Express und Netscape Messenger implementiert.

Zwei wichtige Merkmale kennzeichnen S/MIME:
Privacy: nur der richtige Empfänger kann die Nachricht lesen.
Authentizität: der Empfänger kann sicher sein, dass die Nachricht auch vom notierten Absender stammt. Die Verschlüsselungstechnik verwendet digitale ID´s und nutzt MIME für die Strukturierung von Nachrichten. Eine digitale ID besteht aus einem öffentlichen und einem privaten Schlüssel sowie einer digitalen Signatur.
Die Kombination aus digitaler Signatur und öffentlichem Schlüssel wird als Zertifikat bezeichnet. Bevor verschlüsselte oder digital signierte Nachrichten versendet werden können, benötigt man zunächst eine digitale ID.
Diese bekommt man z.B. bei folgenden Zertifizierungsstellen wie Verisign (verisign.com), Deutsche Post (signtrust.de), Deutsche Telekom (telesec.de) oder bei der Trustcenter GmbH (trustcenter.de).

3.6 PPTP (point-to-point tunneling protocol)

PPTP bezeichnet ein Protokoll für ein Übertragungsverfahren mit Tunneling. Dabei wird über öffentliche Leitungen ein virtuelles Intranet aufgebaut. Es hat sehr hohe Sicherheitsanforderungen, damit der Datenverkehr nicht mitgehört werden kann.

Das PPTP wurde 1996 vom PPTP-Forum entwickelt. Es kommt hauptsächlich in Microsoft-Betriebssystemen zum Einsatz. PPTP ist ausschließlich für den Transfer von IP, IPX und NetBEUI über IP geeignet. Es baut auf den Remote Access Server für Microsoft Windows NT inklusive der Authentifizierung. Wegen der weiten Verbreitung von Windows Clients spielt PPTP beim Aufbau von VPN´s in Microsoft-Netzwerken eine große Rolle.



Die PPTP-Architektur teilt sich in zwei logische Systeme.
Den PPTP Access Concentrator (PAC) und den PPTP Network Server (PNS).
Der PAC verwaltet die Verbindungen und stellt diese zum PNS her. Der PNS ist für das Routing und die Kontrolle der vom PNS empfangenen Pakete zuständig.
Der PAC ist üblicherweise in den Client integriert und stellt eine PPP-Verbindung zum PNS her. Nach der Authentifizierung und Autorisierung wird dem PAC eine IP-Adresse aus dem LAN zugewiesen. Danach beginnt er PPTP-Pakete zu senden.
Die PPP-Rahmen werden mit Generic Routing Encapsulation (GRC) verpackt.
Danach werden die Datenpakete über das IP-Netz zum Ziel transportiert.
Eine Verschlüsselung findet nicht statt. Daher muss bereits bei PPP die Verschlüsselung ausgehandelt werden z. B. mit RC4, was Microsoft auch als Microsoft Point-to-Point-Encryption (MPPE) bezeichnet.

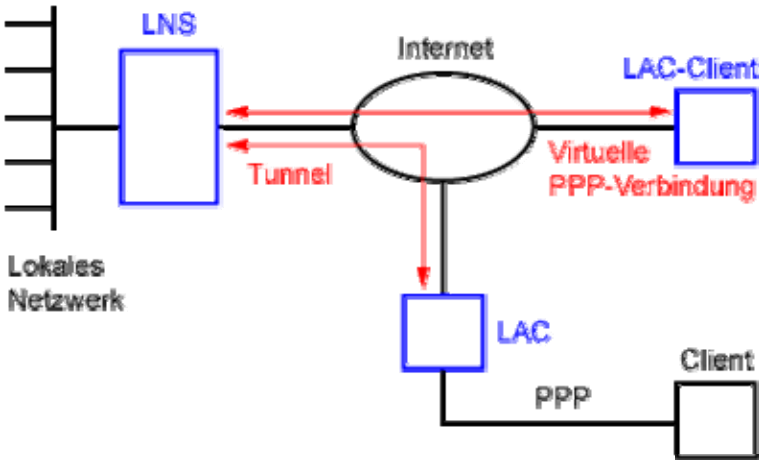
3.7 L2TP (Layer 2 Tunneling Protocol)

Das Layer-2-Tunneling-Protocol hat die Aufgabe PPP-Verbindungen über ein IP-Netzwerk zwischen zwei Netzwerk-Stationen oder zwei eigenständigen Netzwerken herzustellen.
Ein Beispiel wäre z.B. ein Außendienstmitarbeiter, der mit seinem Notebook über eine Wählverbindung Zugang zum Internet hat und darüber eine getunnelte Verbindung zum Netzwerk seiner Firma herstellt, die über eine Standleitung ebenfalls ans Internet angebunden ist. Anstatt einer reellen Punkt-zu-Punkt-Verbindung besteht die Übertragungsstrecke aus mehreren Routern, die miteinander verbunden sind.
Für dieses Szenario gibt es zwei Protokolle:

- L2F - Layer-2-Forwarding
- PPTP - Point-to-Point Tunneling Protocol

Aus diesen beiden Protokollen entwickelte sich das Layer-2-Tunneling Protocol, wobei nahezu alle Anteile aus PPTP in L2TP eingeflossen sind. L2TP bietet jedoch den Vorteil, dass es jedes beliebige Netzwerkprotokoll in den PPP-Rahmen transportieren kann.
PPTP unterstützt jedoch nur IP, IPX und NetBEUI.
L2TP bietet selbst keinen Authentifizierungs-, Integritäts- und Verschlüsselungsmechanismus. Der Schutz der getunnelten Daten muss z.B. mit IPsec erfolgen. In VPN-Lösungen kommt meist eine L2TP/IPsec-Lösung zum Einsatz.

L2TP- Architektur



Die L2TP-Architektur teilt sich in zwei logische Systeme, den L2TP Access Concentrator (LAC) und den L2TP Network Server (LNS). Der LAC verwaltet die Verbindungen und stellt diese zum LNS her. Der LNS ist für das Routing und die Kontrolle der vom LAC empfangenen Pakete zuständig. Das L2TP definiert die Kontroll- und Datenpakete zur Kommunikation zwischen dem LAC und dem LNS. Ein Network Access Server (NAS) stellt einen temporären Zugang für Remote-Systeme zu Verfügung. Der NAS kann im LAC oder im LNS implementiert sein. Es gibt insgesamt zwei Szenarien einen L2TP-Tunnel aufzubauen.

Das erste Szenario sieht eine PPP-Verbindung zwischen dem Client und dem LAC vor z. B. über das Wählnetz (analog oder ISDN). Der LAC tunnelt die PPP-Daten zum LNS und bekommt von diesem eine IP-Adresse aus dem LAN zugeteilt.

Das zweite Szenario sieht eine direkte Unterstützung von L2TP auf dem Client vor. Der Client ist dann selber der LAC. Die Daten werden genauso mit PPP übertragen. Die IP-Adresse aus dem LAN wird auch hier vom LNS zugeteilt. In beiden Fällen ist die Autorisierung und Authentifizierung von den Mechanismen im LAN abhängig. Das ist z. B. über den NAS möglich.

Mit L2TP wird ein Tunnel zwischen LAC und LNS aufgebaut. Der NAS identifiziert den Remote-User über einen Authentifizierungsserver. Ist die Authentifizierung erfolgreich wird der L2TP-Tunnel etabliert. Der LNS identifiziert sich ebenfalls beim Remote-User und bestätigt den L2TP-Tunnel. In diesem Tunnel wird für jede PPP-Verbindung eine Sitzung (session) zwischen LAC und LNS aufgebaut. Mittels des Multiplex-Modus lassen sich in einem Tunnel mehrere Sitzungen aufbauen.

4 Andere Techniken zur sicheren Übertragung

Einweg Hash-Funktionen
Sie sind wie digitale Fingerabdrücke: kleine Datenstücke, die der Identifizierung viel größerer digitaler Objekte dienen. Es sind öffentliche Funktionen (ohne Geheimschlüssel), die mit Hilfe mathematischer Berechnungen einen eindeutigen Wert ergeben.
Hash-Funktionen bieten einen gewissen Grad an Authentifizierung und Integrität an. Sie werden häufig eingesetzt, um die Echtheit eines digitalen Objekts überprüfen zu können. Fast jedes Internet-Protokoll benutzt sie, um Schlüssel zu verarbeiten, eine Folge von Ereignissen zu verketten oder Ereignisse zu authentifizieren. Außerdem ein wichtiger Baustein der digitalen Signatur (s. 4.3 Signaturen).
Heute sind eine ganze Reihe von Einweg-Hash Funktionen in Gebrauch. SHA-1 (Secure Hash Algorithm) ist der Standard Algorithmus, MD4 und MD5 sind weitere bekannte Arten.
Forderungen an eine Hashfunktion h, m sei die Nachricht:

- Der Hashwert h(m) ist leicht zu berechnen.
- Es ist praktisch nicht möglich, zu einem Hashwert h1 eine Nachricht m zu bestimmen, die h(m) = h1 ergibt.

4.1 Pretty Good Privacy (PGP)

PGP (Pretty Good Privacy) ist ein von Phil Zimmermann entwickeltes Programm zur Verschlüsselung von Daten, dessen 1. Version 1991 erschien. Als freie Alternative wurde das Programm Gnu Privacy Guard (GPG) entwickelt, das unter der freien GPL-Lizenz steht.
Hier spielt die asymmetrische Verschlüsselung mit 2 versch. Schlüsseln eine Rolle. Bob erzeugt einen Ver- und einen Entschlüsselungsschlüssel. Seinen Public-Key (Verschlüsselung) kann er veröffentlichen, meist auf auf sog. Key-Servern. Zur Auswahl an Schlüsselarten stehen RSA, ElGamal, DSA oder auch IDEA.
Alice kann diesen Schlüssel finden und mit diesem eine Nachricht an Bob versenden. Bob wiederum kann nun diese verschlüsselte Nachricht mit seinem Private-Key (Entschlüsselung) in Klartext zurückwandeln. Hierbei ist zu erwähnen, dass Bob und Alice sich nicht kennen müssen oder vorher Schlüssel untereinander ausgetauscht haben müssen. Eine verschlüsselte Kommunikation findet hier ohne direkten Kontakt der beiden statt.
Alice kann somit eine Nachricht an Bob verschlüsseln, die nur Bob wieder entschlüsseln kann.

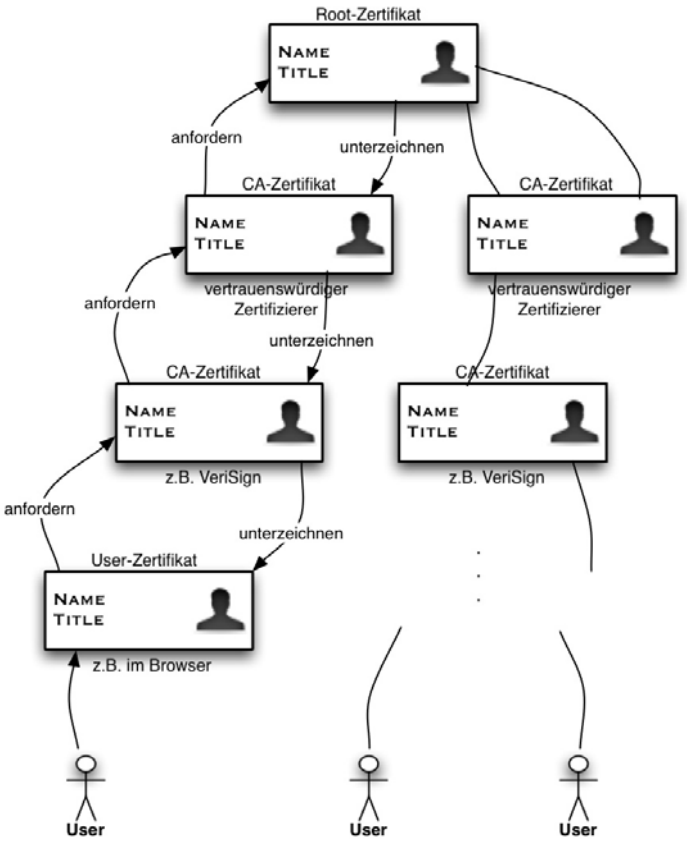
Konkret läuft der praxistaugliche Nachrichtenaustausch noch etwas anders ab. In der Praxis angewandte Systeme setzen auf einen Hybridansatz der symmetrische und asymmetrische Verschlüsselung verwendet, der in der Leistungsfähigkeit begründet ist:
Wenn Alice eine Nachricht an Bob senden möchte, verwendet sie einen symmetrischen Algorithmus, um die Nachricht mit einem Zufallsschlüssel zu verschlüsseln (Sitzungsschlüssel). Sie verschlüsselt diesen Zufallsschlüssel mit Bobs öffentlichem Schlüssel und sendet dann sowohl den verschlüsselten Schlüssel als auch die verschlüsselte Nachricht an Bob
Wenn Bob die verschlüsselte Nachricht und den Schlüssel erhält, macht er das Gegenteil. Er benutzt seinen privaten Schlüssel, um den symmetrischen Zufallsschlüssel zu entschlüsseln und dann den symmetrischen Zufallsschlüssel, um die Nachricht zu entschlüsseln.

4.2 Public-Key-Infrastrukturen (PKI)

Ein Client, der über das Internet mit einem Server eine Verbindung aufnimmt, muss sich ohne vorherige Maßnahmen darauf verlassen können, dass er tatsächlich mit dem gewünschten Server kommuniziert.
Dies wird durch eine von einer vertrauenswürdigen dritten Instanz ausgestellten Bescheinigung (Zertifikat) erreicht, die die Echtheit des Servers bestätigt.
Ein bekanntes Zertifizierungsprotokoll ist der *X.509* Standard.

Das Internet benutzt Zertifikate für viele Protokolle, z.B. IPsec und verschiedene VPN-Systeme, SSL usw. Diese Zertifikate werden Benutzern von einer sogenannten Certificate Authority (CA) ausgestellt. Eine CA kann ein firmenspezifisches Sicherheitsbüro, eine Regierungsstelle oder eine private Firma sein, die sich auf die Ausstellung von Zertifikaten für Internet-Benutzer spezialisiert hat.
Ein Zertifikat enthält den Namen der Person, dessen öffentlicher Schlüssel und die Signatur einer CA. (kurz: Zertifikat = Name + public key + Signatur[CA])
Die CA erzeugt mit ihrem geheimen Schlüssel die Signatur für das jeweilige Zertifikat. Verschlüsselt wird wie meist erst der Hashwert der Inhaberdaten. Überprüft werden kann die Signatur mit dem öffentlichen Schlüssel der CA (Vergleich der Hashwerte).

Diese CAs benötigen ihrerseits Zertifikate (vgl. mit einer hierarchischen Struktur), die wiederum von anderen CAs (z.B. VeriSign) ausgestellt werden. Am Ende der Hierarchieleiter (höchste Ebene) gelangt man zu einigen CAs mit sogenannten Root-Zertifikaten; sie werden von keinem anderen unterzeichnet. Diese „höchsten“ Zertifikate sind in kommerzieller Software (z.B. Browser, VPN-Client…) schon integriert. Das nennt man eine Public-Key-Infrastruktur.



[hierarchische Darstellung einer PKI]

PKIs und CAs bergen eine Fülle von Problemen.
Was bedeutet es, wenn eine CA behauptet, dass sie vertrauenswürdig ist ?
Kann man überprüfen, ob das der Wahrheit entspricht ?
Als Zwischenlösung hatte sich ergeben, dass einige CAs ein zweiteilige Zertifizierungsstruktur geschaffen haben: eine sogenannte Registration Authority (RA), die als Autorität für die Inhalte gilt und die CA selbst nur für dessen Ausstellung verantwortlich ist.
Mittlerweile ist aber das alleinige Modell mit nur einer CA ebenso sicher, als mit einer zweigeteilten RA+CA Struktur. Eine CA kann immer die Inhalte noch verfälschen, sodass eine Aufteilung nutzlos ist.
Alles basiert somit auf dem Vertrauen des Benutzers und der Solidarität und Neutralität der CAs.

Letztlich stellt sich die Frage, wie die CA den Zertifikatsinhaber identifiziert hat. Ob ein Zertifikat nur einen Identifizierer oder eine spezifische Autorisation hält, spielt keine Rolle; die CA muss den Antragsteller identifizieren, bevor sie das Zertifikat ausstellt.
Viele Zertifikatstandards sind auch verbunden mit einer bestimmten Lebensdauer, der CRL, die automatisch das Zertifikat nach einer bestimmten Zeit auf die Liste der ungültigen setzt.
Die meisten Menschen kommen nur über SSL mit PKI in Berührung. In Online-Shops wird über eine SSL-Verbindung ein Zertifikat ausgestellt, das eine autorisierte und gesicherte Verbindung garantiert. Wirklich garantiert ist es jedoch nur, wenn der Online-Shop und das Zertifikat den gleichen Absender haben.

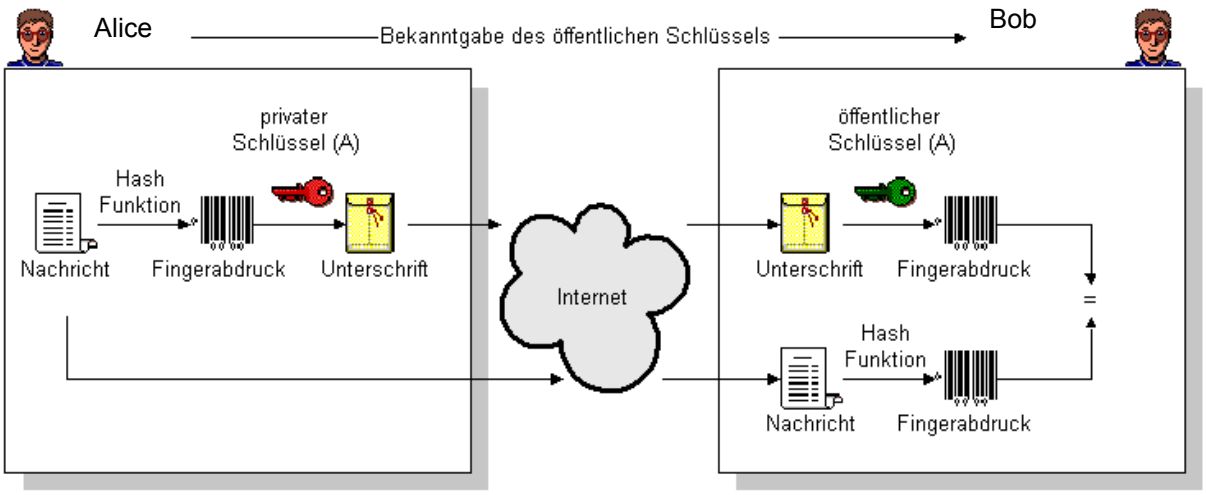
Nach dem Signaturgesetz muss ein Zertifikat folgende Angaben enthalten:

- Name des Inhabers
- Öffentlicher Schlüssel von Inhaber und Zertifizierungsstelle
- Signatur über Name und Schlüssel (= Unterschrift)
- Angaben zu den Algorithmen
- Gültigkeit des Zertifikats
- Fortlaufende Seriennummer des Zertifikats
- Name der Zertifizierungsstelle (CA)
- Angaben über evtl. Begrenzungen auf bestimmte Anwendungen

4.3 Signaturen

Ein weiteres Basiselement sind Signaturen. Sie bieten keinen Vertraulichkeitsschutz, aber sichern Authentifizierung und Integrität. Sie gewährleisten, dass die Nachricht tatsächlich von der Person stammt, von der sie vermeintlich stammt (Authentifizierung), und dass die Nachricht auf dem Weg nicht verändert wurde (Integrität).
Jeder kann dennoch die Nachricht lesen (keine Vertraulichkeit), man kann aber durch eine Prüfung der Signatur eindeutig prüfen, ob die Nachricht möglicherweise durch eine andere Person verändert wurde.
Man kann auch sagen, dass eine digitale Signatur eine mathematische Operation auf einer Menge Bits ist, die nur ein bestimmter Schlüssel ausführen kann.

Konkret läuft eine digitale Signatur einer Nachricht folgendermaßen ab:
Alice wendet eine Einweg-Hash Funktion auf die Nachricht an und signiert den Hash-Wert. Dieses Verfahren der Hash-Wert Signatur bietet wiederum eine Leistungsverbesserung und erhöht die Sicherheit gegenüber einem direkten Signieren der Nachricht.
Sie führt mit der Nachricht und ihrem privaten Schlüssel eine Berechnung durch, um die Signatur zu erzeugen. Die Signatur wird an die digitale Nachricht angehängt. Bob führt eine andere Berechnung mit der Nachricht, der Signatur und der Alices öffentlichem Schlüssel durch, um die Signatur zu prüfen. Eve, die Alices privaten Schlüssel nicht kennt, kann die Signatur überprüfen, kann die Nachricht und eine gültige Signatur nicht fälschen.
Es sind mehrere Algorithmen für digitale Signaturen in Gebrauch: der häufigste ist RSA (s. 2.2.3 RSA). Weitere Vertreter sind DAS (s. 2.2.2 DAS), ElGamal oder Signaturalgorithmen auf der Grundlage elliptischer Kurven (s. 2.2.4 ECC).

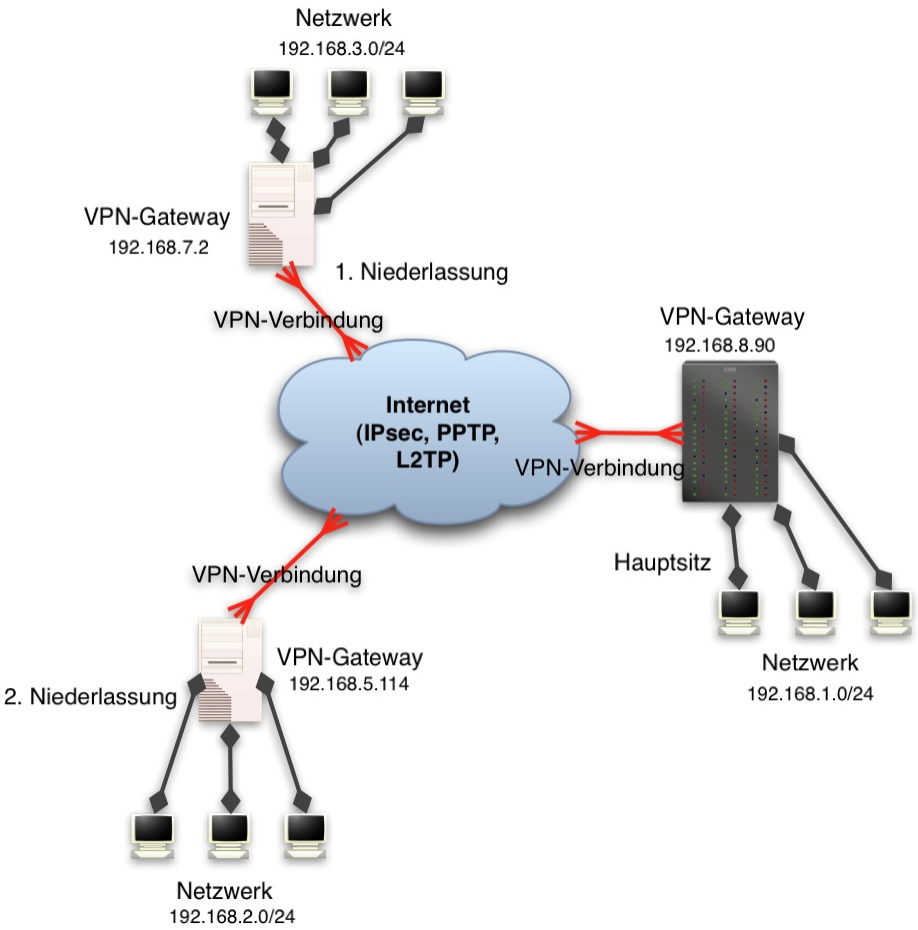


[Austausch einer Nachricht mit einer digitalen Signatur]

4.4 Virtuelle Private Netze (VPN)

Ein virtuelles privates Netzwerk ist einfach eine sichere Verbindung über ein öffentliches Netzwerk mit TCP/IP als Übertragungsprotokoll und genau zwei Kommunikationspartnern.
Man spricht auch von einem sog. Tunnel, in dem IP-Datagramme vollständig gekapselt werden. Dabei werden die Original IP-Pakete an ein anderes Protokoll als sog. Nutzlast (payload) angehängt. Durch diese „Encapsulation“ wird ein zusätzlicher Protokollkopf (Header), der Tunnel-Header, benutzt. Das Generic Routing Encapsulation Protocol (GRE) ist z.B. ein solches Netzwerkprotokoll, das beim PPTP (s. 3.6 PPTP) verwendet wird.
Um nun eine gesicherte Verbindung zwischen zwei Endpunkten aufbauen zu können, müssen auf beiden Seiten der Kommunikationsverbindung zahlreiche Parameter übereinstimmen bzw. aufeinander abgestimmt sein. Die Kommunikationspartner müssen sich zum Beispiel auf die Art der gesicherten Übertragung (Authentifizierung und/oder Verschlüsselung), den Verschlüsselungsalgorithmus sowie die passenden Schlüssel einigen. Auch muss festgelegt werden, wie und wie oft die eingesetzten Schlüssel ausgetauscht werden.
Verschiedene VPNs bieten Sicherheit durch unterschiedliche kryptographische Protokolle. Das üblichste Protokoll ist IPsec, obwohl man immer noch Protokolle vorfindet, die PPTP und L2TP implementieren. Einige VPNs enthalten auch überhaupt keine Kryptographie.
Verschlüsselt werden die IP-Pakete bevor sie getunnelt werden. Dann übernimmt der Zusatzheader des ersten VPN-Gateways die Weiterleitung zum Zielgateway. Er schreibt dabei eine neue Quell- und Zieladresse in den neuen Header. Wenn das Paket am anderen VPN-Gateway ist, wird der Zusatzheader wieder entfernt und das IP-Paket entschlüsselt. Nach Auslesen des Original-Header wird das Paket an den Zielrechner geschickt.

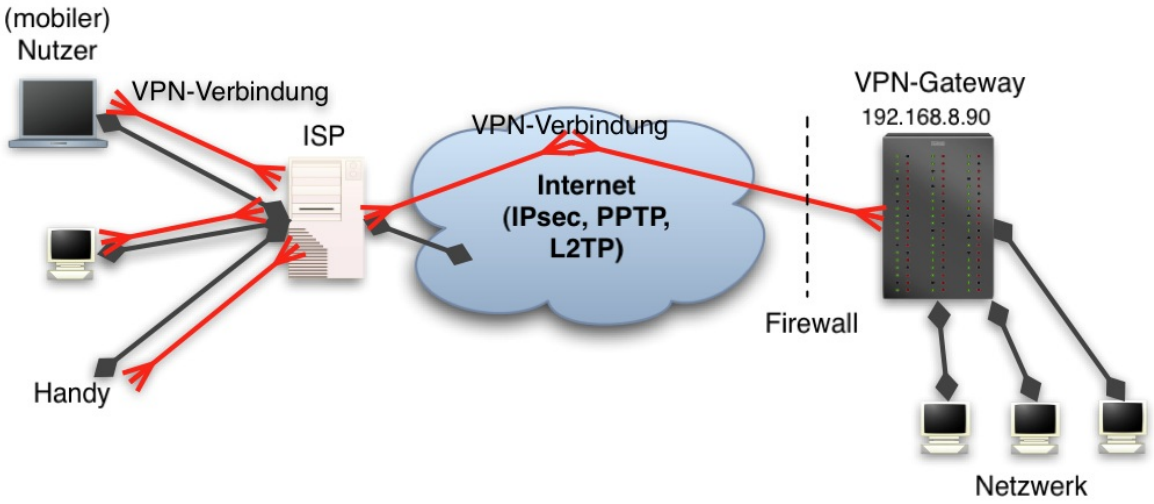
Site-to-Site VPN
Erstens verbinden sie getrennte Teile des gleichen Netzwerks. Ein Unternehmen hat möglicherweise zwei Niederlassungen an verschiedenen Standorten. Jede Niederlassung hat ihr eigenes physisches Netzwerk, und die beiden Netzwerke sind über ein VPN, das über das Internet läuft, miteinander verbunden. Ein VPN ist privater, als eine von der Telefongesellschaft bereitgestellte Lösung.



Daten und Freigaben werden über die VPN-Verbindung wie im Intranet zwischen den Rechnern ausgetauscht.

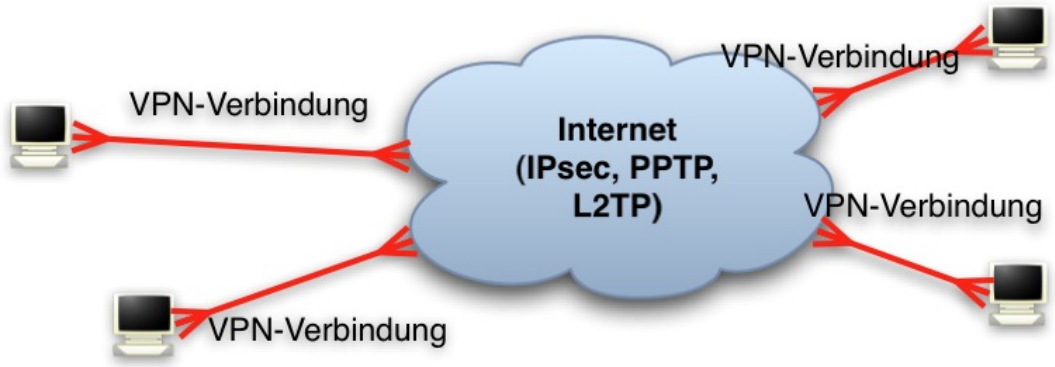
Remote-Access VPN

Zweitens verbinden sie mobile Benutzer, z.B. von zu Hause oder von Hotelzimmern. Es wird über den ISP eine VPN-Verbindung zum VPN-Gateway des Firmennetzwerks aufgebaut.



End-to-site VPN
Auch WLANs werden mittlerweile immer öfter über VPN gesichert. Aufbau vgl. mit Remote-Access VPN. Verbindung läuft hier aber dann über Funk.

End-to-End VPN
Es ist auch möglich, dass ein Tunnel zwischen zwei einzelnen Computern aufgebaut wird. Allerdings erheblicher Verwaltungsaufwand, da jede Kommunikation verschlüsselt wird.



Zu beachten ist hierbei, dass auf jedem der an das VPN angeschlossenen Rechner ein entsprechendes VPN-Protokoll installiert sein muss, da die Arbeitsrechner direkt untereinander und nicht über zwischengeschaltete VPN-Server verbunden werden.

Vorteile
- Die Einrichtung eines VPN ist wesentlich günstiger als eine extra (angemietete) Standleitung
- Technologie in Verbindung mit IPsec gilt als relativ sicher und zuverlässig
- Verschiedene Standorte und Übertragungswege sind flexibel gestaltbar (Kompatibilität)
- Es sind mehrer VPN-Verbindungen über eine Leitung möglich (Flexibilität)

Nachteile
- Konstante Bandbreite ist bei einer Internetverbindung nicht garantiert (Ausfallrisiko)
- Ein schlecht konfiguriertes VPN kann mehr schaden als nützen (Komplexität)
- Regelmäßige Prüfung und Aktualisierung der VPN-Software und deren Schutzmechanismen

5 Kryptoanalyse

Schlüssellänge

Eine der einfachsten Möglichkeiten, kryptographische Algorithmen zu vergleichen, ist die Schlüssellänge. Meist wird ein langer Schlüssel für gut, ein kurzer für schlecht befunden. Die Realität zeigt, dass ein langer Schlüssel nicht automatisch gut sein muss.

Ein kryptographischer Schlüssel ist ein geheimer Wert, durch den ein kryptographischer Algorithmus für jene, die den Schlüssel gemeinsam nutzen, eindeutig wird. Wenn Alice und Bob einen gemeinsamen Schlüssel benutzen, können sie den Algorithmus verwenden, um sicher zu kommunizieren.

Ist ein Schlüssel n Bit lang, gibt es 2^n mögliche Schlüssel. Wenn also ein Schlüssel 40 Bit lang ist, gibt es etwa eine Trillion mögliche Schlüssel. 1998 konnte ein 56Bit langer DES-Schlüssel im Durchschnitt in 4.5 Tagen geknackt werden. (entspricht etwa 90 Milliarden Schlüssel pro Sekunde). Schon 1999 wurde ein Schlüsselsuchprojekt mit Werten von 250 Milliarden Schlüsseln pro Sekunde erreicht.

Empfohlen wird heute meist eine Schlüssellänge von 128 Bit (oder auch 192 und 256 Bit). Diese Angaben beziehen sich auf symmetrische Algorithmen, bei Hash-Funktionen sollten etwa doppelte Längen eingesetzt werden.

Für Algorithmen mit öffentlichen Schlüsseln empfehlen Fachleute 1024 Bit Schlüssel oder sogar noch länger.

5.1 Definition von Kryptoanalyse

Die Kryptoanalyse ist das Gegenstück der Kryptographie. Sie befasst sich damit, aus einem codierten Text den Klartext zu ermitteln. Das kann sowohl direkt als auch über den entsprechenden Schlüssel passieren. Sie wird also eingesetzt um Codes zu knacken.

Es existieren verschiedene Ansätze für eine effiziente Kryptoanalyse:

- Brute-Force
  Siehe 5.2
- Wörterbuch-Attacke (dictionary attack)
  Bekannte Passwörter werden in Reihenfolge ihrer Wahrscheinlichkeit getestet
- Ciphertext Only
  Ermitteln des Schlüssels mit Hilfe mehrerer bekannter codierten Texte
- Probable Plaintext
  Der Geheimtext ist vorhanden sowie eine Vorahnung auf enthaltenen Text
- Known Plaintext
  Geheimtext und Klartext sind bekannt -> Schlüssel kann gesucht werden
- Side Channel
  Neben dem Klartext, dem Chiffrentext und dem Schlüssel spielen die Dauer der Verschlüsselung, zeitlicher Stromverbrauch eines Chips, sowie andere Faktoren eine Rolle um den Verschlüsselungsalgorithmus zu bestimmen.
- Lineare Kryptoanalyse
  Lineare Annäherung an den wahrscheinlichsten Schlüssel
- watermark attack
  Versucht bestimmtes Muster im Klartext nachzuweisen
- Angriff mit frei wählbarem Klartext (chosen-plaintext attack)
- Angriff mit frei wählbarem Geheimtext (chosen-ciphertext attack)
- Adaptive Chosen Plaintext / Differential Cryptanalysis
- Adaptive Chosen Ciphertext
- Chosen Text

Adaptive Chosen Text

5.2 Brute Force

Brute-Force Attacken haben zu 100% Erfolg – allerdings auf Kosten der Zeit. Bei dieser Methode werden einfach alle Möglichkeiten, die in Frage kommen durchprobiert, bis der richtige Schlüssel gefunden wurde. Im schlechtesten Fall muss man also für einen n – langen Schlüssel n Schlüssel testen um auf den richtigen zu stoßen. Weil nur getestet wird, ob ein Schlüssel passt kann die Brute-Force Attacke auf jeden beliebigen Verschlüsselungsalgorithmus angewandt werden.
Brute-Force Attacken verlaufen linear, d.h. die doppelte Anzahl von Computern kann die doppelte Anzahl von Schlüsseln probieren. Die Schwierigkeit einer Brute-Force Attacke steigt aber exponentiell mit der Schlüssellänge. Ist der Schlüssel nur 1 Bit länger, wird eine Attacke um das Doppelte schwieriger/langwieriger. Fügt man zwei Bits hinzu, um das Vierfache schwieriger. Wenn man 10 Bits hinzufügt, ist es ein Tausendfaches schwieriger.

6 Zusammenfassung

Wenn also Kryptographie so mächtig ist, warum kommt es dann zu Sicherheitsproblemen ?
Die Begründung liegt im Unterschied zwischen Theorie und Praxis. Kryptographie ist ein Zweig der Mathematik und die ist theoretisch. Sie folgt logischen unwiderlegbaren Schlussfolgerungen. Aber nur auf dem Papier sehen die vielen Algorithmen gut aus. Meist ist der Faktor Mensch ein großer unberechenbarer Faktor, der jeden noch so ausgeklügelten Algorithmus aushebeln oder zur Sicherheitslücke machen kann. Hardware ist genauso. Sie stürzt ab, verhält sich ab und zu unkorrekt. Auch Software kann zu komplex sein und leidet damit an Überschaubarkeit. Die kryptographische Theorie wird immer mit der Praxis konfrontiert sein, wenn sie in einem konkreten System benutzt wird.

Hierbei kommt der Begriff Entropie recht häufig vor. Entropie ist ein Maß für die Unordnung eines Systems. Insbesondere ist sie im Zusammenhang mit Kryptographie ein Maß für Unsicherheit. Je unsicherer etwas ist, desto mehr Entropie ist darin enthalten. z.B. bedeutet ein Algorithmus der 128 Bit Schlüssel akzeptiert nicht, dass der Schlüssel eine Entropie von 128 Bit besitzt. Es muss nicht erforderlich sein, alle möglichen Schlüssel durchzuprobieren. Die 128 Bit sind lediglich ein Maß des maximalen Arbeitsaufwands, der erforderlich ist, um den Algorithmus zu überwinden und den Schlüssel aufzudecken. Sie sagen nichts über das Minimum aus.
Viele Schlüssel werden aus Passwörtern erzeugt. Ein System, das 10 Zeichen lange ASCII-Passwörter akzeptiert, erfordert für die Darstellung vielleicht 80 Bit, hat aber eine Entropie von viel weniger als 80 Bit (Qualität des Schlüssel). Bitfolgen, die höheren ASCII-Werten entsprechen, erscheinen bei diesem System überhaupt nicht, und Passwörter, die echte oder fast echte Wörter sind, kommen viel eher als zufällige Zeichenketten vor. Wenn man von einem 128 Bit Schlüssel ausgeht, bräuchte man ein 32 Zeichen langes Passwort. (man geht von einer Entropie bei Passwörtern von 4 Bit/Zeichen aus)
Auch wenn der Verschlüsselungsalgorithmus den Schlüssel nicht optimal benutzt, reduziert dies effektiv die Entropie des Schlüssels (Qualität des Verschlüsselungsalgorithmus).

7 Quellangaben

7.1 Literatur

[1] Bruce Schneier – Secret & Lies

7.2 Internet

[2] www.google.de
[3] www.wikipedia.de
[4] http://www.faqs.org/rfcs/rfc2003.html
[5] www.tele-task.de
[6] http://home.nordwest.net/hgm/krypto/deploy.htm
[7] http://www.datenschutzzentrum.de/selbstdatenschutz/internet/index.htm
[8] www.lfd.niedersachsen.de/
[9] http://www.bsi.bund.de/
[10] http://www.itl.nist.gov
[11] www.itwissen.info

7.3 Software

[12] www.cryptool.de

7.4 Sonstiges

[13] GDV2 Skript – Harald Sorber