1. WAS IST SSL

SSL (Secure Socket Layer) ist ein Übertragungsprotokoll, das verschlüsselte Verbindungen über Transportprotokolle wie zum Beispiel TCP/IP ermöglicht. Der Vorteil von SSL liegt vor allem darin, dass es unabhängig vom Transportprotokoll ist. So kann zum Beispiel über Telnet mit SSL verschlüsselt eine sichere Verbindung zwischen Server und Client aufgebaut werden.

Das Haupteinsatzgebiet ist die Verschlüsselung des Webprotokolls HTTP (HTTP-S). Weitere Einsatzgebiete sind SSH (Secure Shell) und S-FTP(Secure FTP).

Bei SSL kommen verschiedene Algorithmen zur Anwendung, die die eigentliche Verschlüsselung vornehmen (siehe Abschnitt 3).

1.1. Protokollaufbau:

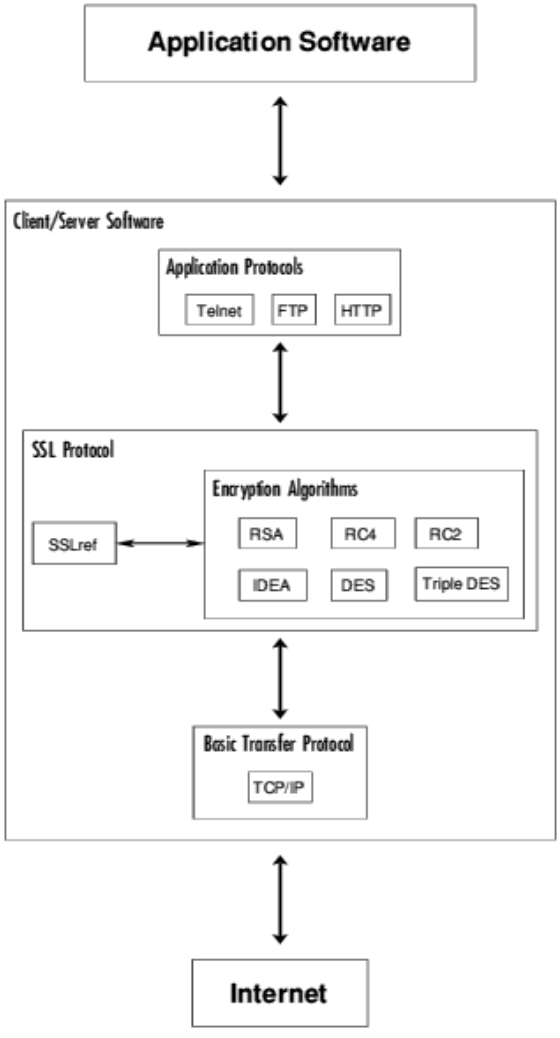

Abbildung 1: SSL-Protokollaufbau

Wie bereits erwähnt setzt SSL auf einem beliebigem Transportprotokoll auf. Daten, die von einem Applikationsprotokoll wie zum Beispiel HTTP, Telnet, FTP, ... an eine darrunterliegende Schicht des ISO/OSI-Netzwerkmodells gereicht werden, passieren die SSL-Record-Schicht und durchlaufen dort einen im Handshake ausgemachten Verschlüsselungsalgorithmus.

Die so verschlüsselten Daten werden über das Transportprotokoll an das Netzwerk weitergereicht und durchlaufen beim Empfänger dieselben Stationen nur in umgekehrter Reihenfolge.

Das SSL-Record-Protokoll besteht aus Paketen, die jeweil einen Header und einen Body besitzen. Im Body steht neben den Daten der Message Authentication Code, der die Echtheit der eigentlichen Daten bestätigen soll. Im Header stehen Metadaten über den Body, wie zum Beispiel Größe und der verwendete Verschlüsselungsalgorithmus. Eine Aufgabe des SSL-R-P ist es, den Datenstrom am Sender zu fragmentieren und zu komprimieren und beim Empfänger wieder zusammenzusetzen.

2. PROTOKOLLABLÄUFE

2.1. Handshake

Jede neue SSL-Verbindung beginnt mit einem Handshake.

- Der Client schickt eine Anfrage zum Verbindungsaufbau mit einer Liste der von ihm unterstützten Verschlüsselungsprotokolle im Klartext an den Server.
- Der Server schickt eine Klartextantwort zurück, die seine digitale ID und das von ihm gewählte Verschlüsselungsprotokoll enthält.
- Der Client generiert einen Sitzungsschlüssel, der für die Dauer der Verbindung gültig ist und schickt diesen mit dem öffentlichen Schlüssel des Servers verschlüsselt zurück.
- Der Server schickt eine Kopie der bisherigen Übertragung mit dem Sitzungsschlüssel verschlüsselt an den Client. Dieser schickt das gleiche Paket an den Server.
- Die Verbindung ist dann zustande gekommen, wenn die zwei Pakete identisch sind.
- Danach beginnt der verschlüsselte Datenaustausch.

Dieser Ablauf ist der einfachste SSL-Handshake und enthält noch keine Authentifizierung. Bei Authentifizierung werden die öffentlichen Schlüssel aller Teilnehmer an Hand öffentlich zugänglicher Authentifizierungsserver, die Kopien der Schlüssel zu den dazugehörigen Rechnern besitzen, bezogen.

2.2. Datenaustausch

Für den Datenaustausch wird ein symmetrischer Verschlüsselungsalgorithmus verwendet. Der Schlüssel sowie der Algorithmus werden im Handshake ausgemacht.

3. VERSCHLÜSSELUNGSALGORITHMEN

SSL verwendet mehrere Verschlüsselungsalgorithmen – asymmetrische für Authentifizierung und symmetrische für den Datenaustausch. Die hier beschriebenen Algorithmen sind die wichtigsten im SSL-Protokoll vorkommenden.

3.1. DES/3DES

3.1.1. Was ist DES?

DES ist ein an Altersschwäche leidender Verschlüsselungsalgortithmus, der Mitte der 70er zum Data Encryption Standard erkoren wurde. DES basiert auf einer Entwicklung von IBM mit Namen Lucifer.

3.1.2. Schlüssellängen

Der DES-Schlüssel ist 64 Bit lang, wobei jedes achte Bit für Paritätszwecke verwendet wird. Daraus ergibt sich eine effektive Schlüssellänge von 56 Bits.

Nach heutigem Stand der Technik sind Schlüssel dieser geringen Längen nicht mehr sicher genug, um damit sinnvolle Verschlüsselung zu betreiben.

3.1.3. Konstruktionsprinzip

DES ist ein Blockchiffrieralgorithmus und basiert auf Permutationen und Substitutionen. Der originale DES-Algorithmus beinhält 16 Runden, eine Initial Permutation und eine Final Permutation die von 64-Bit langen Blöcken durchlaufen werden. Die Initial Permutation, die die Umkehrfunktion der Final Permutation darstellt, ist eine reine Permutation. Die Bits werden an Hand einer Tabelle verwürfelt.

Danach erfolg eine 2-Teilung der Daten. Eine Hälfte wird mit der sogenannten Expansion Mutation auf 48Bit aufgeblasen und mit dem ersten Schlüssel, der ein 48-Bit kurzer (sic) Teilschlüssel des eigentlich 64Bit langem DES-Schlüssel darstellt, verschlüsselt. Jede DES-Runde wird ein neuer Teilschlüssel ermittelt.

Die eigentliche Verschlüsselung erfolgt mit sogenannten S-Boxes, deren Funktionsweise lange Zeit unklar war. S-Boxes sind Tabellen mit 16 Spalten und 4 Reihen. Ein 6 Bit langer Input wird kodiert, in dem das erste und das letzte Bit zsammengenommen die Reihe und die mittleren 4 Bits die Spalte angeben. Diese 6 Bits werden durch das 4 Bit lange Zeichen der S-Box Tabelle ersetzt.

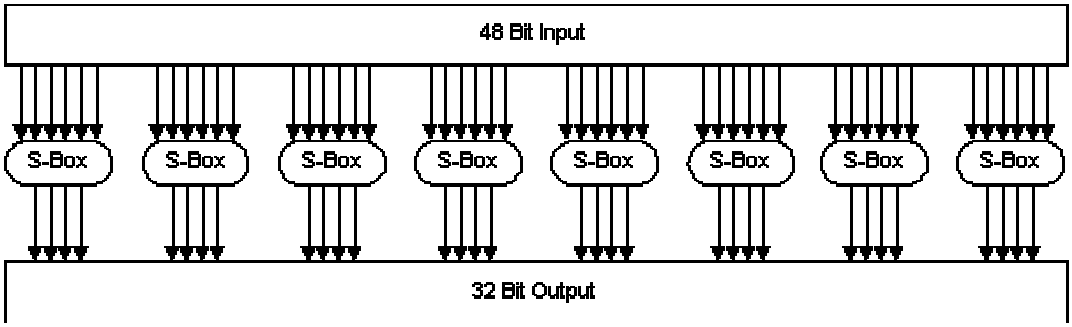

Abbildung 2: S-Box Verschlüsselung

Der Aufbau der S-Boxes ist das Um und Auf der Sicherheit von DES.

Nach der S-Box durchläuft das Signal eine P-Box, die eine weitere Verwürfelung der Bits darstellt, aber nicht unmittelbar zur Sicherheit von DES beiträgt.

3.1.4. Was ist 3DES? – Unterschied zu DES

Der Unterschied zwischen DES und 3DES (oder auch Triple DES) ist, dass 3DES eine 3-fache DES Verschlüsselung mit 3 verschiedenen Schlüsseln darstellt. Dadurch erhöht sich die Anzahl der möglichen Schlüssel von 2^{56} auf 2^{168}. Somit ist eine wesentlich höhere Sicherheit gegeben, die nach heutigem Standard für zivile Zwecke immer noch ausreicht.

3.2. DSA

DSA ist das Akronym für Digital Signature Algorithm und wurde 1991 von NIST (National Institute of Standards and Technology) als neuer Standard für Digitale Signaturen (DSS) vorgeschlagen. Der Ernennung von DSA als neuen Standard folgte ein Sturm der Entrüstung. Einerseits wurde RSA bereits als defacto Standard für digitale Signaturen verwendet und andererseits wurde DSA von der NSA entwickelt, was die Vermutung zulässt, dass diese eine Hintertür eingebaut haben. Ein weiterer Nachteil gegenüber RSA ist die Geschwindigkeit – die Verifikation der Signatur kann bei DSA um bis zum 40-fachen langsamer sein. Die Schlüsselgenerierung ist auf jeden Fall bei DSA schneller; hier ist die Geschwindigkeit aber eher irrelevant, da Schlüssel selten neu generiert werden. Das Hauptargument gegen DSA war aber die

vorgesehene Schlüsselgröße – 512 Bits waren für einen neuen Standard nicht genug. Als Reaktion darauf wurde die Schlüssellänge variabel gestaltet – von 512 Bits bis 1024 Bits.

3.2.1. Wie funktioniert DSA?

DSA ist eine Variante des Schnorr und ElGamal Signaturalgorithmus und verwendet folgende Parameter:

p	Eine 512-1024 Bit lange Primzahl
q	Ein 160 Bit langer Primfaktor von $(p-1)$
g	$h^{(p-1)/q} \bmod p$, wobei h eine beliebige Zahl kleiner als $p-1$, sodass $h^{(p-1)/q} \bmod p$ größer als 1 ist.
x	Eine beliebige Zahl kleiner als q
y	$g^x \bmod p$

Der Algorithmus verwendet die One-way Hash Function $H(m)$ (siehe SHA), die Parameter p,q und g sind öffentlich. x ist der private Schlüssel, y ist der öffentliche Schlüssel.

Um eine Nachricht m zu signieren, müssen folgende Schritte durchlaufen werden.

(1) Alice generiert eine Zufallszahl, die kleiner als q ist.

(2) Alice generiert ihre Signatur:
$r = (g^k \bmod p) \bmod q$
$s = (k^{-1}(H(m) + xr)) \bmod q$
Die Parameter r und s sind ihre digitale Signatur, sie sendet diese an Bob.

(3) Bob verifiziert die Signatur:
$w = s^{-1} \bmod q$
$u_1 = (H(m) * w) \bmod q$
$u_2 = (rw) \bmod q$
$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$
Wenn gilt $v=r$, dann ist die Signatur verifiziert.

3.3. MD5

3.3.1. Was ist MD5?

MD5 ist eine verbesserte Variante von MD4, einer Hash-Funktion. MD5 produziert Hashes der Länge 128 Bit.

3.3.2. Was ist eine Hash-Funktion?

Eine Hash-Funktion ist eine Ein-Weg Funktion, d.h. trotz bekanntem Algorithmus und Ergebnis lässt sich der ursprüngliche Wert nicht mehr errechnen. Eine Hash-Funktion muss eine Ein-Weg-Funktion und Kollisionsresistent sein.

3.3.3. Was ist Kollisionsresistenz?

Eine Kollision tritt dann auf, wenn zwei oder mehr Inputs den selben Hash ergeben. Mathematisch ausgedrückt:

$$f(x)=f(z), \forall x \neq z$$

Die Funktion f ist hierbei die Hashfunktion, x und z sind unterschiedliche Ausgangswerte für die Funktion.

Ist diese Gleichung gegeben, tritt eine Kollision auf.

3.3.4. Wie funktioniert MD5

MD5 verarbeitet die Nachricht zu je 512 Bit, wobei diese 512 Bit in 16 32Bit-Blöcke aufgeteilt werden. Das Produkt des Algorithmus ist ein Quadrupel von 32Bit Blöcken, die zusammen einen 128 Bit Hash bilden.

Der erste Schritt ist, die Nachricht auf ein ganzzahliges Vielfaches von 512 Bit aufzublasen, wobei ein Eins und soviel Nullen angehängt werden, dass 64 Bit auf das ganzzahlige Vielfache noch bleiben. Diese 64 Bit werden mit einer Repräsentation der Länge der ursprünglichen Nachricht aufgefüllt.

Nach diesem Schritt werden 4 sogenannte Kettenvariablen (`A,B,C,D`) initialisiert. Danach kann der eigentliche Algorithmus beginnen:

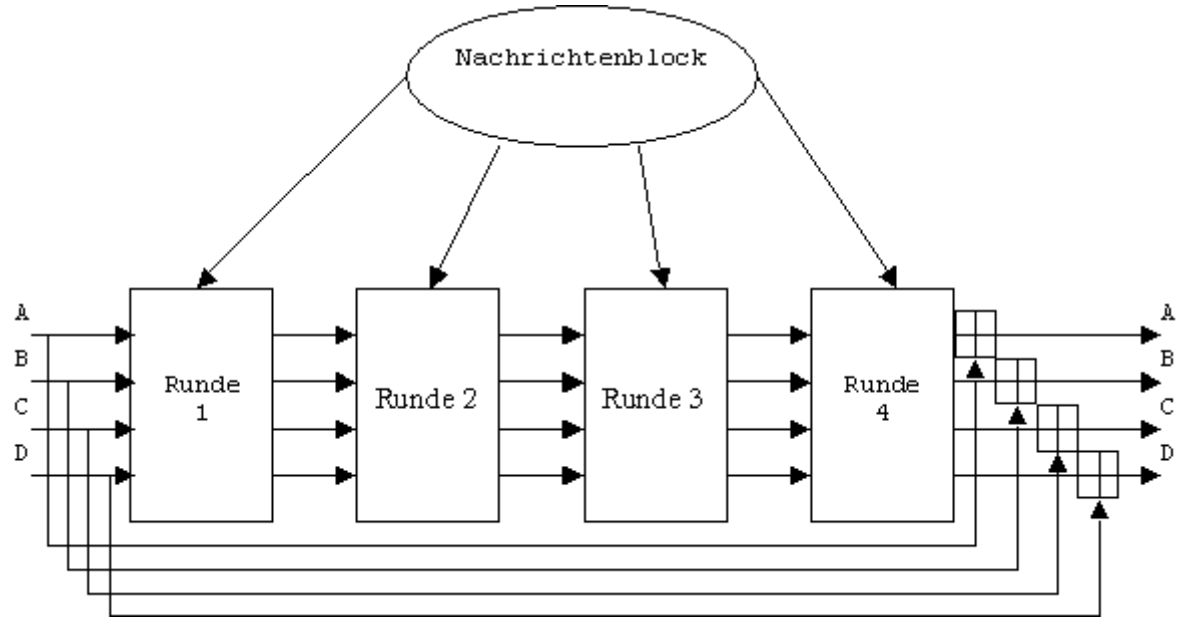

Abbildung 3: MD5 Hauptschleife

Der Algorithmus hat 4 Runden, die alle sehr ähnlich sind. Jede Runde wendet eine eigene Operation 16 mal auf einen Nachrichtenblock an. Jede Operation wendet eine nicht-lineare Funktion auf drei Kopien der Variablen A,B,C,D an. Dann wird das Resultat zur Kopie der vierten Variable addiert, ein Unterblock aus Text und einer Konstante. Danach wird das Resultat

eine variable Anzahl von Bits nach rechts Rotiert und das Ergebnis zu einer der Kopien von A,B,C oder D addiert. Dieses Resultat ersetzt dann eine der vier Variablen.

Es gibt insgesamt 4 nicht-lineare Funktionen, je eine pro Runde.

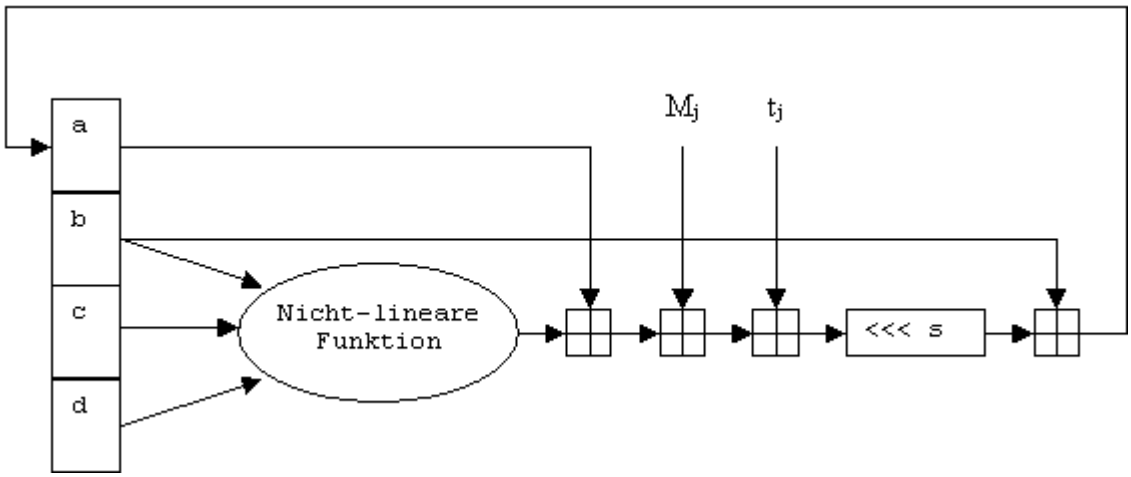

Abbildung 4: MD5 Operation

Diese Abbildung zeigt eine MD5 Operation, bei der 3 Variablen in einer nicht-linearen Funktion verwürfelt, mit der 4. addiert, mit einem Subblock der Nachricht M_j und einer Konstanten t_j verwürfelt und eine variable Anzahl von Bits nach rechts rotiert werden, bevor eine der Variablen hinzuaddiert wird und das Ergebnis eine der ursprünglichen Variablen ersetzt.

3.3.5. Kollisionsresistenz von MD5

Die Kollisionsresistenz des 128Bit Hash von MD5 liegt bei 2^{64} Versuchen, was nach heutigem Stand der Technik zu gering bzw. die Wahrscheinlichkeit einer Kollision eher zu hoch ist.

3.4. RC2/RC4

3.4.1. Was ist RC2

RC2 ist ein 1997 von Ron Rivest entwickelter Algorithmus, der DES als Verschlüsselungsstandard ablösen sollte. RC2 arbeit, gleich wie DES, mit 64-Bit Blöcken, die durch MIX- und MASH-Runden laufen und mit dem jeweiligen Rundenschlüssel verschlüsselt werden.

3.4.2. Key Expansion

Der jeweilige Rundenschlüssel durchläuft eine eigene Prozedur. Ein Teil des Schlüssels wird vom Benutzer vorgegeben, der Rest anhand eines Algorithmus aufgefüllt. Dieser Wert ist 128 Bit lang und wird als Index in einer PITABLE verwendet. Der Wert, der in der PITABLE an der Stelle Index steht ist der Wert, der als Rundenschlüssel verwendet wird. Die PITABLE ist so gewählt, dass möglichst keine Abhängigkeiten zwischen zwei verschlüsselten Blöcken entstehen können.

3.4.3. Verschlüsselung

Die Verschlüsselung durchläuft 5 MIXING-Runden, eine MASHING-Runde, sechs MIXING-Rungen, eine MASHING-Runde und noch mal fünf MIXING-Runden. Die Dekodierung ist die umgekehrte Kodierung, durchläuft also die gleichen Runden in umgekehrter Reihenfolge.

3.4.4. Was ist RC4?

RC4 ist ein Datenstromchiffrieralgorithmus, der mit variabler Schlüssellänge arbeitet. Er wurde 1987 von Ron Rivest für RSA Security, Inc. entwickelt und blieb für 7 Jahre ein proprietärer Algorithmus. 1994 gelangte der Sourcecode in die Cypherpunks-Mailinglist und von dort aus über Usenet und FTP in die gesamte Welt.

3.4.5. Der Algorithmus

RC4 hat einen relativ einfachen, aber effektiven Algorithmus:

Der Schlüssel muss unabhängig vom Datenstrom sein. RC4 hat eine 8x8 S-Box: S_0, S_1, … S_{255}. Die Werte der S-Box sind Permutationen der Zahlen 0 bis 255, wobei die Permutation eine Funktion des Schlüssels mit variabler Länge ist. Um ein variables Byte zu erzeugen, wird der folgende Algorithmus durchlaufen (die Zähler i und j werden jeweils mit 0 initialisiert):

```
i = (i + 1) mod 256
j = (j + Si) mod 256
swap (Sj, Si)
t = (Si + Sj)
K = St
```

Das Byte K wird mit dem Klartext XORed um den Chiffriertext bzw. der Chiffriertext mit K XORed um den Klartext zu erzeugen.

Die Initialisierung der S-Box ist auch nicht schwieriger. Zuerst werden alle S_0, S_1, ... S_{255} linear von 0 bis 255 befüllt. Danach wird ein weiteres 256-Bytelanges Array K_0, K_1, ... K_{255} erzeugt und mit dem Schlüssel befüllt. Falls notwendig, wird der Schlüssel solange wiederholt, bis das gesamte Array vollständig initialisiert ist. Danach wird der Zähler j mit 0 initialisiert und folgender Algorithmus ausgeführt:

```
for i = 0 to 255
      j = (j + Si + Ki) mod 256
      swap(Sj, Si)
```

Und das war's. Der erzeugte Output ist im höchsten Grade unkorreliert und nichtlinear und somit immun gegenüber differentieller und linearer Kryptoanalyse.

3.5. RSA

3.5.1. Was ist RSA?

RSA ist ein asymmetrischer Verschlüsselungsalgorithmus, der nach seinen Erfindern Ron Rivest, Adi Shamir und Leonard Adleman benannt ist. Asymmetrische Kryptologie verwendet öffentliche und private Schlüssel. Eine mit dem öffentlichem Schlüssel verschlüsselte Nachricht ist nur mit dem privaten Schlüssel lesbar (bzw. umgekehrt).

RSA beruht auf Primfaktorenzerlegung, die für heutige Computer nur sehr schwer bewältigbar ist.

3.5.2. Schlüssellängen

RSA – Schlüssel können unterschiedliche Längen besitzen, von 1024 Bit bis zu 4096 Bit sind zur Zeit gebräuchliche Längen.

3.5.3. Konstruktionsprinzip

Ein sogenannter Modulus `n` wird aus dem Produkt zweier großer Primzahlen `p` und `q` errechnet:

$$n = p * q$$

Danach wird eine Nummer `e` ausgewählt, die kleiner als `n` und relativ Prim zu `(p-1) * (q-1)` ist. Dann wird eine Nummer `d` gesucht, so dass `k = e*d - 1` ein Divisor von `(p-1) * (q-1)` ist. Eine Art `d` zu finden ist der euklidische Algorithmus:

`ggT(a,b)` ausrechnen für `a > b > 0`:

$$\begin{aligned} a &= q_0 * b + r_0 \\ b &= q_1 * r_0 + r_1 \\ r_0 &= q_2 * r_1 + r_2 \\ &\dots \\ r_k &= q_{k+2} * r_{k+1} + r_{k+2} \end{aligned}$$

Um eine Nachricht `m` zu verschlüsseln, wird folgende Formel angewandt:

$$c_i = m_i^e \bmod n$$

Das Entschlüsseln erfolgt entsprechend mit der Funktion:

$$m_i = c_i^d \bmod n$$

Der Beweis dazu:

$$\begin{aligned} & c_i^d \bmod n = \\ = & (m_i^e)^d \bmod n = m_i^{de} \bmod n = \\ = & m_i^{k*(p-1)*(q-1)+1} \bmod n = \\ = & m_i * m_i^{k*(p-1)*(q-1)} \bmod n = \\ = & m_i * 1 \bmod n \\ = & m_i \bmod n \end{aligned}$$

3.6. SHA-1

3.6.1. Was ist SHA-1?

SHA steht für Secure Hash Algorithm und ist ein Standard für Digitale Signaturen. SHA wurde von der NSA entwickelt.

3.6.2. Welches Konstruktionsprinzip hat SHA-1?

SHA produziert Hashes der Länge 160 Bit.

Der erste Schritt ist die ursprüngliche Nachricht auf ein ganzzahliges Vielfaches von 512 Bits Länge aufzublasen. Dies geschieht durch das sogenannte `Padding`, wobei eine Eins und danach so viele Nullen, wie noch benötigt werden um eine Länge von `k*512 - 64` Bit zu erreichen, an

die Nachricht angehängt werden. Danach wird noch eine 64 Bit lange Repräsentation der Länge der Nachricht angehängt.

Im nächsten Schritt werden fünf 32-Bit Variablen initialisiert. Jetzt kann der eigentliche Algorithmus beginnen:

Es werden in einer Schleife Blöcke von 512 Bit verarbeitet bis zum Ende der Nachricht. Jeder Schleifendurchlauf hat vier Runden mit je 20 Operationen. Jede Operation beinhält eine nicht lineare Funktion auf drei der fünf Variablen, die bei jeder Runde auf den Nachrichtenblock angewendet werden (vgl. MD5).

3.6.3. Wie (un)wahrscheilich sind Kollissionen bei SHA-1?

Die Kollisionsresistenz von SHA-1 ist mit 2^{80} Versuchen deutlich höher als bei MD5, und reicht nach heutigem Stand der Technik aus, um von einem Kollisionsresistenten Algorithmus zu sprechen. Mittlerweile sind SHA-196, SHA-256 und SHA-512 mit deutlich geringeren Kollisionswahrscheinlichkeiten in Arbeit.