Sicherheit

Philipp Adler

25. Oktober 2015

Inhaltsverzeichnis

1 Grundlagen Securityverfahren

1.1 Verschlüsselungsarten

1.2 Kommunikationsszenarien

1.3 Sicherheitsziele

1.4 Bedrohungsszenarien

2 Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung reicht bis in die Antike. Damals wussten nur die Adressierten von dem Geheimnis, nach welchem Verfahren die Botschaft verschlüsselt wurde. Cäsar zum Beispiel verschob jeden Buchstaben um 4 Stellen. So wurde aus Hallo Kdoor. Aus diesem Verschlüsselungsalgorithmus entstanden zum einen Blockchiffren, auf den ich in dem folgenden Kapitel näher eingehen werde und die Stromchiffren.[1]

2.1 Blockchiffre

Blockchiffren teilen die Nachricht, die verschlüsselt werden soll, in eine fixe Anzahl an Blöcken, die entweder 64 oder 128 Bit groß sind. Typische, bekannte Blockchiffre sind Data Encryption Standard, Advanced Encryption Standard und International Data Encryption Algorithm.[1]

2.1.1 Der Data Encryption Standard - DES

"DES wurde 1977 vom amerikanischen 'National Institute of Standards and Technologies (NIST)' veröffentlicht." [1] Bei diesem Verfahren wird eine Blocklänge von 64 Bit und ein DES-Schlüssel von 56 Bits plus 8 "Parity Check Bits" [1] eingesetzt. Die ersten 56 Bits werden immer zufällig generiert, wobei die letzten 8 Bits dafür sorgen, dass keine Übertragungsfehler auftreten. Da 56 Bit zufällig sind, können daraus 2^{56} Schlüsseln erzeugt werden. 16 Runden wird ein Block in einen 64 Bit großen Ausgabeblock umgewandelt. Bei jedem Durchgang wird ein anderer Schlüssel für die Verschlüsselung angwendet. [1][2]
Das Schema
Beim Verschlüsselungsverfahren wird der Klartext in Blöcke umgewandelt, welche alle jeweils eine Länge von 64 Bit haben, Eingangspermutation IP. Dieser Block wird dann nochmals zerlegt, sodass daraus 2 mal 32 Bit Blöcke entstehen. Der Data Encryption Standard besteht aus 16 Runden. In jeder Runde wird auf die rechte Hälfte ein Verschlüsselungsalgorithmus f angewendet.
f ist in unserem Fall die Rundenfunktion, welche aus 56 zufälligen Bits 48 Bits auswählt. Diese werden mit den 32 Bit der rechten Hälfte, die auf 48 Bit expandiert wurden, mittels XOR-Gatter verknüpft. Die 32 Bit Blöcke werden in 4 aufgeteilt und bekommen zusätzlich am Rand die Nachbarbit.

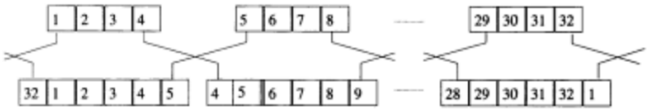

Abbildung 1: Expansionsabbildung des DES [3]

"Die resultierenden 48 Bits werden in acht Blöcke zu je sechs Bits aufgeteilt" [3], welche als Input für das S-Boxen angewandt werden. Die Substitution-Box besteht aus einer 4 * 16 Matrix, "wobei in jeder Zeile eine Permutation der Zahlen von 0,...,15 steht." [3] Die beiden Randbit der Blöcke entscheiden die Zeile und der Rest, die inneren Bit der Blöcke, die Spalte der Substitution-Box. Die ausgewählte Zahl wird dann binär als 4 Bit Block angegeben.[3]

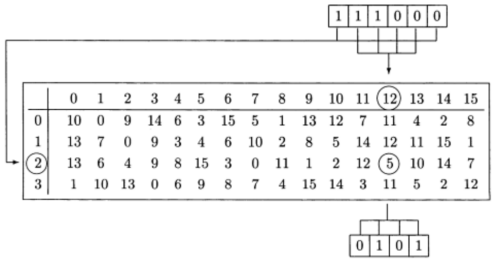

Abbildung 2: S-Box[3]

Da wir nun wieder acht Blöcke zu je 4 Bit haben, können diese zu 32 Bit zusammengefasst werden.[3]

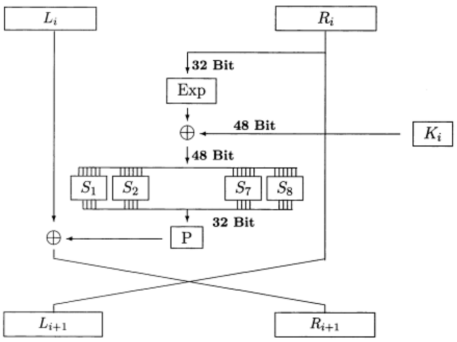

Abbildung 3: Die DES-Rundenfunktion [3]

Das daraus resultierende Ergebnis wird nochmals permutiert und bitweise mit einem XOR-Gatter mit der linken Hälfte verknüpft und "bildet die rechte Seite der neuen Runde." Unter permutiert versteht man, dass jedes einzelne bit als Zahl dargestellt wird und mit 8 addiert wird.[3]

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 9 & 17 & 23 & 31 & 13 & 28 & 2 & 18 & 24 & 16 & 30 & 6 & 26 & 20 & 10 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \\ 8 & 14 & 25 & 3 & 4 & 29 & 11 & 19 & 32 & 12 & 22 & 7 & 5 & 27 & 15 & 21 \end{pmatrix}$$

Abbildung 4: Permutation de DES [3]

Da der 48 Bit Schlüssel, welcher vom 56 Bit-Hauptschlüssel hergeleitet wird, bei jeder Runde ein anderer ist, muss dieser irgendwie generiert werden. Der Hauptschlüssel wird permutiert und die Funktion PC-1 teilt diesen in 2 Blöcke zu je 28 Bit. Bei der Permutierung werden die 8 Paritätsbit entfernt, die Bits mit der Nummer 8, 16, 24, 32, 40, 48, 56, 64, wobei nur noch 56 Bit übrig bleiben. Jede der beiden Hälften wird bei jeder Iteration zirkulärisch links für die Verschlüsselung, nach rechts für die Entschlüsselung gesshiftet. Das heißt, dass jeder Block entweder ein oder zwei Bit nach links rotiert und auf 24 Bit extrahiert wird. So kann es nicht vorkommen, dass ein Rundenschlüssel nicht zweimal angewendet wird.
"Nach 16 Runden werden die 64 Bit einer Ausgangspermutation unterzogen" [3], woraus der Geheimtext resultiert. Die Ausgangspermutation ist die inverse von der Eingangspermutation, dass heißt alle 64 Bit Blöcke werden zu einem Geheimtext zusammengeführt. Nachteil dieser Implementierung ist, dass die Ein- und Ausgangspermutation öffentlich sind und so von Angreifern berechnet werden können, was die Sicherheit drastisch verringert. [3][2]

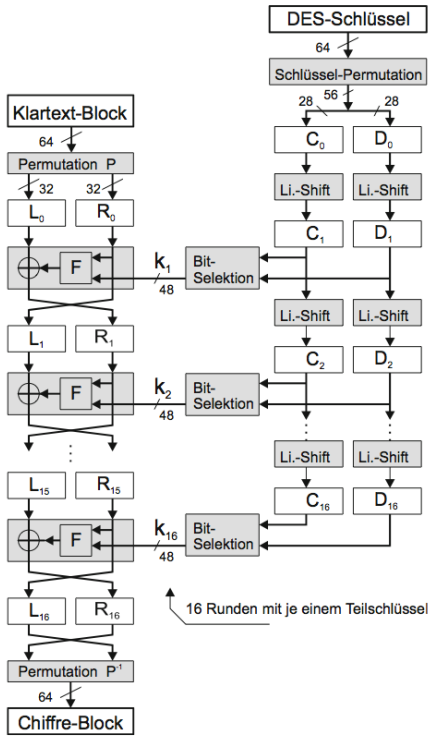

Abbildung 5: DES-Verschlüsselung-Schema [4]

2.1.2 Der Advanced Encryption Standard - AES

Da der DES-Algorithmus aus einen verhältnismäßig kurzen 56-Bit Schlüssel besteht und dieser 1999 durch einen sogenannten Brute-Force-Angriff in 22 Stunden geknackt wurde, müssen andere Vorschläge her. Die Alternative hieß AES, Advanced Encryption Standard, ist ebenfalls eine symmetrische Block-Chiffre, mit einer Blocklänge von 128 Bit. [4]

Das Schema

Der Unterschied zum DES ist, dass AES eine flexible Block- und Schlüssellänge besitzt. Der Standard besitzt eine standardisierte Blocklänge von 128 Bit und Schlüssellängen von 128 Bit, 192 Bit und 256 Bit. Wieviele Runden absolviert werden hängt von der Schlüssellänge ab. 10 Runden bei einer Schlüssellänge von 128 Bit, was derzeit der Standard ist, 12 Runden bei 192 Bit und 14 Runden bei 256 Bit. "Vor der ersten Runde wird ein Rundenschlüssel mit dem Klartext XOR-verknüpft."[3] [4]

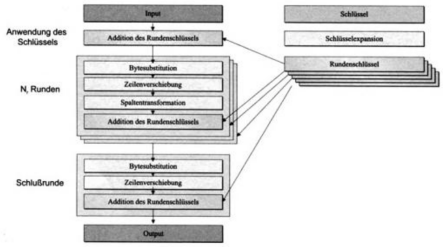

Abbildung 6: Schema des AES [3]

Beim AES wird der Text und die Ergebnisse als Bytes in eine 4x4-Matrix, in sogenannte States, gespeichert. Die Einträge erfolgen spaltenweise, wobei von links nach rechts angeordnet wird. Bei dieser Transformationsfunktion werden die 128 Bit in 16 Bytes geteilt. Diese Bytes werden als States in der Matrix bezeichnet. Die Suche erfolgt mittels der Indexe. [3][4]

d_0	d_4	d_8	d_{12}
d_1	d_5	d_9	d_{13}
d_2	d_6	d_{10}	d_{14}
d_3	d_7	d_{11}	d_{15}

↓

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Abbildung 7: Datenstruktur: ein State [4]

Wie auch DES eine Rundenfunktion besitzt, hat AES ebenfalls eine. Diese besteht aus SubBytes, ShiftRow, MixColumn und AddRoundKey.

Beim SubBytes werden die States substituiert. Zuerst wird das Eingangsbyte durch die gebildete Inverse $a_{r,c}$ ersetzt. "Entscheidend für das Verständnis des AES ist, dass die Bytes als Elemente des Körpers $GF(2^8)$ aufgefasst werden."[3] Dadurch ist es möglich Bytes zu addieren und multiplizieren. Dieser Körper wird durch Polynome in Form als

$GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ dargestellt. Für die Berechnung wird das Byte in Bits dargestellt und einem Polynom zugeordnet. Es wird dann nicht mehr mit Bytes gerechnet sondern mit Polynomen.
10100010 = $x^8 + x^6 + x$ Der neue State wird mit einer 8x8 Matrix mod 2 multipliziert. Daraus ergibt sich ein Byte großer Ergebnisvektor der zum Schluss mit der Konstante GF(2^8) addiert wird. Vorteil an diesem Algorithmus ist das die Bildung der Inverse eines Bytes nichtlinear ist, was die Analyse des AES erschwert.[3][4]

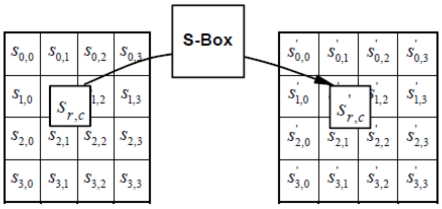

Abbildung 8: Die Abbildung SubBytes [3]

ShiftRow und MixColumn stellen die Permutation der Rundenfunktion dar. Wobei ShiftRow zeilenweise operiert und MixColumn spaltenweise. Beim ShiftRow besteht die Permutation aus einem Linksshift, welcher von der Blocklänge abhängig ist. Ausgenommen ist die erste Zeile, weil diese nie verschoben wird. Ansonsten wird die 2.Zeile um C1 Bytes, die 3.Zeile um C2 und die 4.Zeile um C3 Bytes verschoben. Falls die Blocklänge 128 Bit beträgt, wird wie folgt nach Links geshiftet.[3]

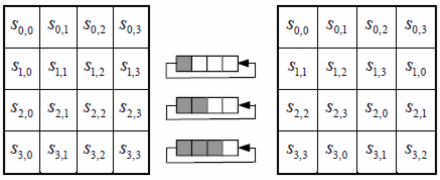

Abbildung 9: Die Abbildung ShiftRow [3]

Wie schon oben erwähnt kümmert sich der MixColumn um die Spalten, welche aus 4 Byte bestehen, die wie bei SubBytes als Polynom dargestellt werden. Da wir hier nur 4 Byte haben, ist der Grad kleiner gleich 3. Das daraus erzeugte Polynom wird mit einer Konstante mod ($x^4 + 1$) multipliziert.[3]

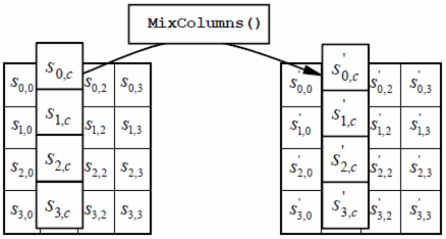

Abbildung 10: Die Abbildung MixColumn [3]

AddRoundKey wird vor Beginn der Runden und am Ende jeder Runde eingesetzt. Um den aktuelle Rundenschlüssel und der Block mittels einer XOR-Verknüpfung addiert. Der Rundenschlüssel wird durch die Formel (Rundenanzahl + 1) * (Anzahl der Wörter pro Matrix) hergeleitet. Da bei 128 Bit mehr Bits zur Verfügung stehen als benötigt werden, muss der Schlüssel expandiert werden. Bei der Expansion wird der externe Schlüssel in seine Bestandteile zerlegt, was im Falle von 128 Bit 4 Wörter sind. Dabei wird i und i+3 mittels XOR verknüpft.
In jeder neuen Runde werden andere Rundenschlüssel verwendet, indem man in der Ersten die ersten vier Wörter und in den nächsten die nächsten vier verwendet.[3]

2.1.3 IDEA (International Data Encryption Algorithm)

Der International Data Encryption Algorithm ist ebenfalls wie DES und AES eine symmetrische Block-Chiffre, welche aus einer Blocklänge von jeweils 64 Bit und einer Schlüssellänge von 128 Bit besteht.[4]
Das Schema
Das IDEA besteht aus 9 Runden, wobei die ersten 8 Runden alle den gleichen Vorgang ausführen. Der Klartext bestehend aus 64 Bit wird in 4 Blöcke aufgespalten. Der erste und letzte Block wird mit einem Teilschlüssel modulo $(2^{16} + 1)$ multipliziert, weil da immer ein Rest auftretet, da es sich um eine Primzahl handelt. Die inneren Blöcke werden stattdessen modulo (2^{16}) addiert. In der Mitte des Schemas werden 2 andere Teilschlüssel eingesetzt, welche die Ergbnisse von oben addieren oder multiplizieren. Am Ende jeder Runde werden die innen Ausgänge vertauscht. Da in jeder Runde ein anderer Klartext zum Einsatz kommt, werden auch andere Teilschlüssel angewandt. “ Aus den Primärschlüssel werden 52 Teilschlüssel von 16 Bit Länge erzeugt, von denen je sechs in acht Runden zum Einsatz kommen.“[6] Da die ersten 8 Runden nun beschrieben wurden, kommen wir zur letzten. Bei der 9ten Runden wird der Hauptteil des Verschlüsselungsschemas ausgelassen, dagegen werden 4 Teilschlüssel $k_{9,1}$, $k_{9,2}$, $k_{9,3}$, $k_{9,4}$ eingesetzt.[4][6]

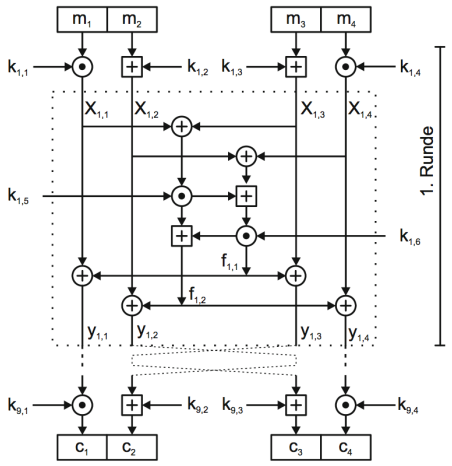

Abbildung 11: IDEA Schema [4]

Die Entschlüsselung funktioniert wird die Verschlüsselung nur in andere Richtung, nur die Teilschlüssel k9,x werden invertiert. [4]

2.2 Der RSA-Algorithmus

Eines der bekanntesten Public-Key-Verfahren ist der RSA-Algorithmus. Entwickelt wurde es 1977 von Rivest, Shamir, Aleman. Das Verfahren beruht auf der Schwierigkeit der Faktorisierung von Zahlen, der sogenannte Satz von Euler. Beim RSA wird ein Nachrichtenblock als Zahl interpretiert, welche kleiner als modula n ist. [3]

2.2.1 Schlüsselerzeugung

Für die Erzeugung des Schlüssels n benötigt man das Produkt von zwei Primzahlen p und q, die mindestens 512 Bit lang sind 2^{512} und geheim gehalten werden. Die Wahrscheinlichkeit, dass es sich um eine Primzahl handelt ist 2^{-9}. Um das faktorisieren zu erschweren gibt spezielle Kriterien. Erstens die Primzahlen sollten sich nicht sehr unterscheiden, aber nicht beieinander liegen. Für die Teilfremdezahl sollte man möglichst kleine Teiler haben.
Als nächstes wird eine Zahl e, enciphering, die teilfremd zu $\varphi(n) = (p - 1)(q - 1)$ ist, ausgewählt. Unter Teilfremd versteht man, dass die beiden Zahlen keinen gemeinsamen Teiler haben. Mit diesen beiden Zahlen kann der öffentliche Schlüssel (e, n) gebildet werden. Damit der Empfänger die Nachricht entschlüsseln kann, benötigt er einen privaten Schlüssel d. Die Zahl d wird mit Hilfe des Euklidischen Algorithmus $e * d \bmod (p - 1)(q - 1) = 1$ berechnet.

RSA-Signatur

"Mit Hilfe des RSA-Verfahrens kann man auch leicht eine digitale Signatur realisieren: Man bildet zunächst den Hashwert $h = hash(m)$ der zu signierenden Nachricht m, und berechnet die Signatur, indem man diesen Hashwert 'entschlüsselt'."[1] Um die Signatur zu prüfen, wird der Hash des Dokuments nochmal gebildet und " dann der Wert

sig durch Potenzierung mit dem öffentlichen Schlüssel e 'entschlüsselt' wird.“[1]. Wenn beide übereinstimmen war die Signatur korrekt. [1][3][6]

2.2.2 Verschlüsseln

Bei der Verschlüsselung wendet der Sender den öffentlichen Schlüssel des Empfängers an seinen Nachrichtenblock an.
$c = m^{\mathrm{e}} \pmod{n}$

2.2.3 Entschlüsseln

Der Empfänger muss nur noch seinen privaten Schlüssel auf den Geheimtext c zum Entschlüsseln anwenden.
$m = c^{\mathrm{d}} \pmod{n}$

3 SSL/TLS-Protokoll

3.1 SSL/TLS Grundlagen

3.2 SSL/TLS im Protokollstapel

3.3 Client-Server-Kommunikation

3.4 Das SSL-Handshake

3.5 TLS asymetrisch

3.6 TLS symetrisch

3.7 TLS hybrid

3.8 TLS Algorithmen

4 Schwierigkeiten bei Software

4.1 Buffer Ovrflow

4.2 OpenSSL

Abbildungsverzeichnis

Literatur

[1] Jörg Schwenk. *Sicherheit und Kryptographie im Internet.* Springer 4.Auflage, 2014.

[2] Maarten van Steen Andrew S. Tanenbaum. *Verteilte Systeme.* Pearson 2.Auflage, 2008.

[3] Thomas Schwarzpaul Albert Beutelspacher, Heike B. Neumann. *Kryptografie in Theorie und Praxis.* Vieweg 1.Auflage, 2005.

[4] Joachim Swoboda Stephan Spitz, Michael Pramateftakis. *Kryptografie und IT-Sicherheit.* Vieweg 2.Auflage, 2011.

[5] Thomas Wilke Ralf Küsters. *Moderne Kryptographie.* Vieweg 1.Auflage, 2011.

[6] Lars Packschies. *Praktische Kryptographie unter Linux.* München: Open Source Press, 2005.

[7] Albrecht Beutelspacher. *Kryptologie.* Springer, 2015.

[8] Dipl.-Ing. Thomas Toth. *Improving Intrusion Detection Systems.* 2003.

[9] Mahbod Tavallaee Ali A. Ghorbani, Wei Lu. *Network Intrusion Detection and Prevention.* Springer, 2010.

[10] Wolfgang Ertel. *Angewandte Kryptographie.* Hanser 4.Auflage, 2012.

[11] Dietmar Wätjen. *Kryptographie: Grundlagen, Algorithmen, Protokolle.* Spektrum 2.Auflage, 2008.