



Kryptographie

data encryption standard

Versuche nichts zu verbergen, denn die Zeit,
die alles hört und sieht, deckt es doch auf.
-Sophokles

Kryptographische Verschlüsselung mithilfe des
DES-Verfahrens und die Übersetzung eines Textes
durch ein selbstgeschriebenes Delphi-Programm

Inhaltsverzeichnis

<u>Thema:</u>	<u>Seite:</u>
<u>1.</u> Terminologie S.3	
<u>1.1</u> Steganographie	3
<u>1.2</u> Kryptoanalyse	3
<u>1.3</u> Kryptographie	3/4
<u>2.</u> Symmetrische Verschlüsselungsverfahren	4
<u>3.</u> Rechenarten	5
<u>3.1</u> Modulo Rechnung	5
<u>3.2</u> XOR-Verknüpfung	6
<u>4.</u> Die Geschichte des DES-Verschlüsselungsverfahrens	7
<u>5.</u> Die Funktion des DES-Verschlüsselungsverfahrens	8
<u>5.1</u> Eine Runde	9
<u>5.2</u> Eingangspemutation/Ausgangspemutation	10
<u>5.3</u> Die Funktion f	11
<u>5.4</u> Die 8 S-Boxen	13
<u>5.5</u> Die Expansionspemutation	14
<u>5.6</u> Die P-Box-Pemutation	14
<u>5.7</u> Die Rundenschlüssel	15
<u>5.8</u> Die Kompressionspemutation	16
<u>6.</u> Beispiel	17-20
<u>7.</u> Anhang	21
<u>7.1</u> Schlüsselaustausch mit symmetrischen Verschlüsselungsverfahren	21/22
<u>7.2</u> Die Länge des Schlüsselraums und die heutige Rechenleistung	22
<u>8.</u> Literaturverzeichnis	23

1. Terminologie

Um näher auf das Thema der Kryptographie eingehen zu können, möchte ich erst einmal die Grundlegenden Begriffe behandeln. Bei der Kryptologie handelt es sich um den Teil der Mathematik, der sich mit der Ver- und Entschlüsselung bzw. mit dem Verbergen von Daten beschäftigt. Diese lässt sich in drei Unterkategorien aufteilen:

1.1 Steganographie

Dieser Teil der Mathematik befasst sich mit dem Verbergen von Daten. Die Verwendung dieser Methode findet sich beim Verstecken von Daten in den niederwertigen Bits der Farbinformationen von Bildern wieder. Auch in Audio-Dateien lassen sich auf Grund des Rauschens gut Daten verstecken.

1.2 Kryptoanalyse

Die Kryptoanalyse befasst sich mit der Entschlüsselung bzw. mit dem Knacken von verschlüsselten Daten, ohne die Kenntnis des zur Codierung verwendeten Schlüssels.

1.3 Kryptographie

Kryptographie stammt aus dem Griechischen. Das Wort Kryptographie lässt sich teilen in KRYPTOS und GRAPHEIN, was soviel wie „geheim“ und „schreiben“ bedeutet. Sie befasst sich folglich mit der Geheimhaltung bzw. mit der Verschlüsselung von Nachrichten / Daten. Das System der Verschlüsselung kann in Form einer Funktion dargestellt werden, bei dem M für Nachricht (engl. message), auch Klartext genannt, E für verschlüsseln (engl. encryption), D für Entschlüsseln (engl. decryption) steht. Der Chiffrentext (verschlüsselter Text) wird mit C und der Schlüssel mit K (engl. Key) gekennzeichnet. Oberflächlich betrachtet ergibt sich somit die Funktion zur Ver- und Entschlüsselung:

Verschlüsseln: $E_k(M) = C$

Entschlüsseln: $D_k(C) = M$

Kombination: $D_k(E_k(M)) = M$

Hierbei ist zu beachten, dass die Methoden, mit denen verschlüsselt wird, unterteilt werden. Zum einen in symmetrische Verschlüsselungsverfahren, zum anderen in asymmetrischen Verschlüsselungsverfahren. Weiterhin unterscheidet man bei der Kryptographie zwischen Stromchiffren und Blockchiffren. Bei den Stromchiffren werden die einzelnen Buchstaben nacheinander verschlüsselt und bei den Blockchiffren werden die Daten in Blöcke (z.B. in 64Bit-Blöcke) eingeteilt und anschließend als Ganzes verschlüsselt. Diese Facharbeit soll die Kryptographie, anhand eines ausgewählten Beispiels aus den symmetrischen Verschlüsselungsverfahren näher erläutern.

2. Symmetrische Verschlüsselungsverfahren

Bei den symmetrischen Verschlüsselungsverfahren werden identische Schlüssel (engl. key) zur Ver- und Entschlüsselung verwendet. Da die meisten Algorithmen bekannt sind, mit dem die Verschlüsselungen arbeiten, hängt die gesamte Sicherheit des Kryptosystems von dem verwendeten Schlüssel ab. Aus diesem Grund werden die Schlüssel bei den symmetrischen Verschlüsselungsverfahren auch „secret keys“ genannt. Die Algorithmen sind allgemein bekannt, damit erfahrene Personen der Kryptoanalyse die sog. Stärke des Kryptosystems testen können. Denn wenn es nicht gelingt, ein Kryptosystem erfolgreich in einer hohen Geschwindigkeit zu knacken, wird das Vertrauen in das verwendete System von der Gesellschaft gestärkt. Ein weiteres Problem stellt sich mit der Übermittlung des Schlüssels dar. Denn es stellt sich die Frage, wie beim symmetrischen Verfahren der verwendende Schlüssel sicher vom Sender zum Empfänger gelangt. Auf dieses Problem wird weiter im Anhang eingegangen (Abschnitt 7.1).

3. Rechenarten

3.1 Modulo Rechnung

Bei der ganzzahligen Division einer natürlichen Zahl a durch eine natürliche Zahl b bleibt ein ganzzahliger Rest r [$0 < r < (b - 1)$]. Zum Beispiel:

$$\begin{array}{lcl} 34 : 7 = 4 & \text{Rest } 6 & \text{oder} & 34 = 7 \cdot 4 + 6 \\ 15 : 5 = 3 & \text{Rest } 0 & \text{oder} & 15 = 5 \cdot 3 + 0 \end{array}$$

Man definiert deshalb:

Seien a und b zwei natürliche Zahlen ($b \neq 0$), so findet man zwei eindeutige natürliche Zahlen q und r , für die gilt:

$$a = q \cdot b + r \text{ mit } 0 \leq r < |b|$$

r ist dann der Rest von a nach der Teilung durch b . Man schreibt:

$$a \bmod b = r \quad (a \text{ modulo } b = r)$$

Aus dieser Definition geht hervor, dass für negative a, b auch ein positiver Rest r entsteht. Dies bedeutet, dass die Ausgangszahlen nicht eindeutig zurückführbar sind, da:

$$(-31) : (-4) = 7 \quad \text{Rest } 3$$

aber:

$$(-31) \neq 7 \cdot (-4) + 3$$

Beispiel:

$$648 : 19 = 34 \quad \text{Rest } 2$$

$$\begin{array}{r} \underline{57} \\ 78 \\ \underline{76} \\ 2 \end{array}$$

also:

$$648 \bmod 19 = 2$$

3.2 XOR-Verknüpfung

Bei der XOR-Verknüpfung (aus dem engl. e**X**clusive-**OR**) handelt es sich um eine binäre Rechnung, die modulo verwendet.

Wenn $a, b \in \{0,1\}$, dann gibt es ein $c \in \{0,1\}$, dass genau dann $c=1$ ergibt, wenn entweder $a=1$ oder $b=1$ ist:

$$a \text{ XOR } b = c$$

Mit modulo geschrieben ergibt sich daraus:

$$(a + b) \bmod 2 = c$$

Beispiel:

1 XOR 0 = 1	oder	(1 + 0) mod 2 = 1
0 XOR 1 = 1	oder	(0 + 1) mod 2 = 1
1 XOR 1 = 0	oder	(1 + 1) mod 2 = 0
0 XOR 0 = 0	oder	(0 + 0) mod 2 = 0

Zur Vereinfachung der Schreibweise wird im Folgenden statt Addition modulo 2, XOR verwendet.

4. Die Geschichte des DES-Verschlüsselungsverfahrens

¹Im Jahre 1973 gab das National Bureau of Standards (NBS) der USA, heute National Institute of Standards and Technology (NIST), in einer öffentlichen Ausschreibung den Anstoß zur Entwicklung eines einheitlichen Verschlüsselungsstandards. Daraufhin wurde von Horst Feistel im Auftrag von IBM der Algorithmus namens LUCIFER vorgestellt, der auf 64Bit-Blöcken mit einem 128Bit-Schlüssel basiert. Dieser arbeitet mit mehrfacher Wiederholung von abwechselnder Substitution und Permutation der einzelnen 64Bit-Blöcke, mit verschiedenen Teilen des Schlüssels. Bis heute gibt es eine Menge abgewandelter Verfahren, die auf diesem Prinzip aufbauen. Aus diesem Grund werden sie auch als Feistel-Chiffren bezeichnet. Als eine einfache Weiterentwicklung von LUCIFER, wurde DES im Jahre 1977 veröffentlicht und in den USA offiziell eingeführt. Als Konsequenz daraus, konnten Kryptographieexperten laufend die Sicherheit von DES prüfen, damit die Anwender immer über die Zuverlässigkeit und Sicherheit Bescheid wussten. Lange Zeit galt DES als sicher und bietet auch heute noch einen gewissen Schutz vor den meisten Angreifern. Die Sicherheit des Verfahrens und der verwendete Schlüsselraum wird im Anhang weiter erläutert (Abschnitt 7.2).

¹ Q1 , S.53f

5. Funktion des DES-Verschlüsselungsverfahrens

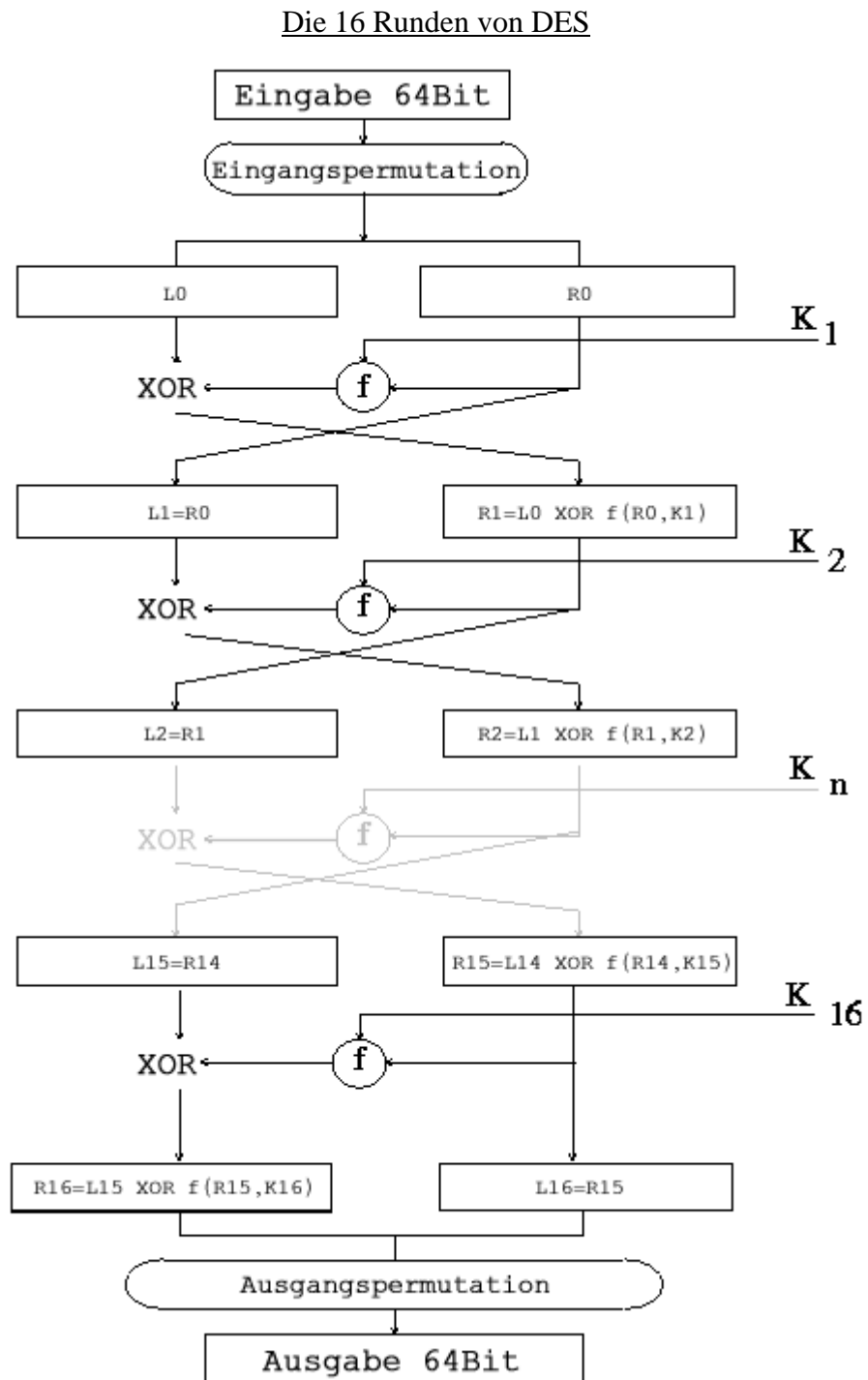


Abbildung 1: eigene Abbildung nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003², S.58

5.1 Eine Runde

Beim DES-Verfahren wird der Text erst einmal in 64Bit-Blöcke eingeteilt. Dieser 64Bit-Block durchläuft erst einmal die Eingangspemutation (invers zur Ausgangspemutation). Nach der Eingangspemutation wird der Block in zwei Hälften geteilt (jeder 32Bit), einmal in L0 und in R0. Der R0 Block wird dann auf eine bestimmte Weise mit einem bestimmten Rundenschlüssel substituiert und daraufhin per XOR mit L0 verknüpft. Dieser mit XOR verknüpfte Teil wird dann zu R1 und der ursprüngliche Teil R0 wird zu L1. Nun hat der ursprüngliche 64Bit-Block eine von 16 Runden durchlaufen. Wenn nun die Hälften bei L16 und R16 angekommen sind, werden sie wieder zu einem 64Bit-Block zusammengefügt, allerdings wird R16 als linker Teil und L16 als rechter Teil verwendet. Bevor der Chiffretext fertig ist, durchläuft der Block noch einmal die Ausgangspemutation.

Eingangspemutation

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Ausgangspemutation

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Abbildung 2: eigene Abbildungen nach

5.2 Eingangspermutation/Ausgangspermutation

Die Eingangspermutation vertauscht die Bit-Positionen des 64Bit-Blocks mit Hilfe einer vorgegebenen S-Box (= Substitutions-Box). Diese S-Box enthält Zahlen von 1 bis 64, wobei jede Zahl nur einmal vorkommt. Zum Beispiel rutscht bei der vorliegenden S-Box Bit 58 auf Bitposition 1, Bit 50 auf Bitposition 2,.... Die Ausgangspermutation ist invers zur Eingangspermutation, was bedeutet, dass sie die voraus gegangene Eingangspermutation umkehrt. Obwohl die Eingangs- bzw. Ausgangspermutationen eigentlich überflüssig sind, da sie keinen Einfluss auf die Sicherheit des Verfahrens nehmen, besitzen sie dennoch einen Sinn. Sie dienen dazu den Klartext und Chiffretext byteweise in den DES-Chip (DES war ursprünglich für den Einsatz in Hardware entworfen worden) zu laden. Es war so konzipiert, dass jede ausgehende Nachricht automatisch durch einen sog. DES-Chip verschlüsselt wird, und beim Empfänger durch seinen DES-Chip wieder entschlüsselt wird.

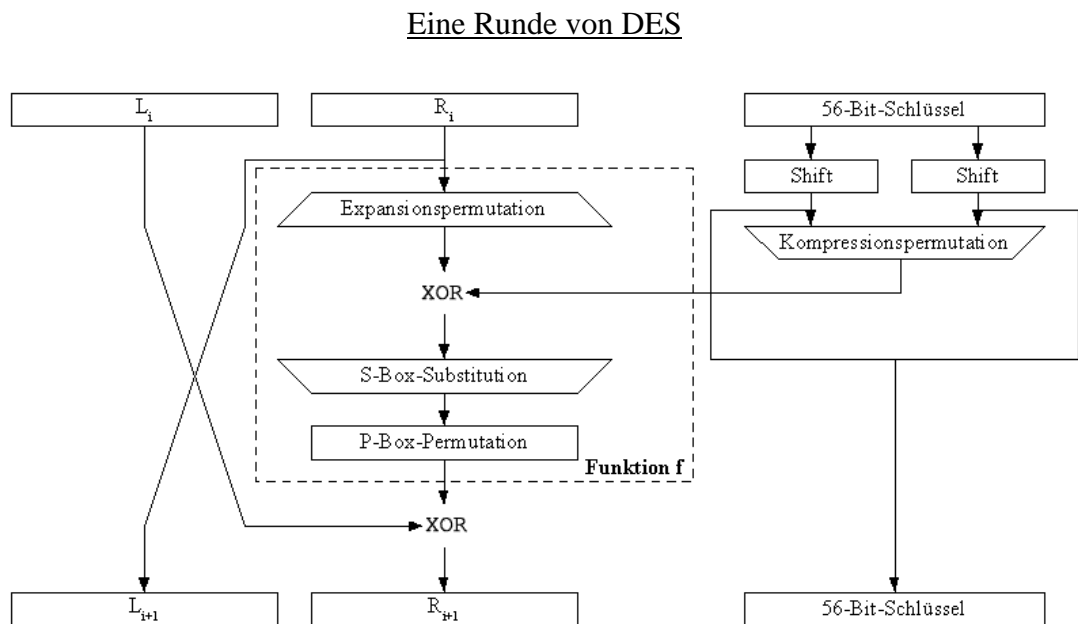


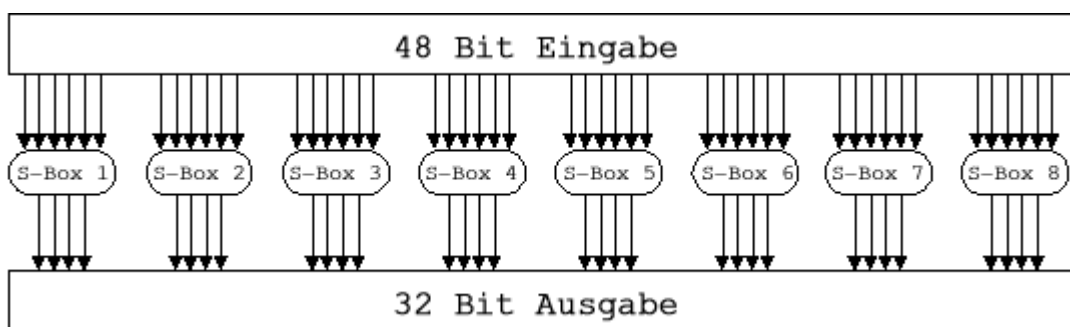
Abbildung 3: eigene Abbildung nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003 ², S.60

5.3 Die Funktion f

Wie schon vorher erwähnt wird die Verschlüsselung eines Klartextes durch ein „Rundensystem“ erreicht. Dabei wird in der aktuellen Runde i zuerst der 32Bit lange R_i Teil mit Hilfe der Expansionspermutation auf 48Bit aufgeweitet (Abschnitt 5.5). Daraufhin werden diese 48Bit per XOR mit dem 48Bit langem Rundenschlüssel verknüpft. Nun folgt der mit Abstand wichtigste Teil der DES-Verschlüsselung, die S-Box-Transformation. Zunächst werden die 48Bit in acht 6Bit-Blöcke aufgeteilt, wobei je ein 6Bit-Block als Eingabe für eine der acht konstanten S-Boxen dient. Das jeweils erste und letzte Bit dieses 6Bit-Blockes, als Zahl interpretiert, bestimmt die Zeile in der zugehörigen S-Box und jeweils das 2,3,4 und 5Bit des Blockes, als Zahl interpretiert, bestimmen die Spalte der S-Box. Bei dem Eintrag, der an dieser Stelle der S-Box gefunden wird, handelt es sich um eine Zahl von 0 bis 15, die als 4Bit-Binärzahl ausgegeben wird. Alle S-Boxen zusammen bilden also eine 32Bit-Ausgabe, die noch einmal durch die konstante P-Box permutiert wird. Nach der Permutation wird der 32Bit-Block XOR mit L_i verknüpft und es entsteht R_{i+1} . Der ursprüngliche R_i Teil wird nun zu L_{i+1} und eine neue Runde wird durchlaufen. (Beispiel im Abschnitt 8)

Die S-Box Transformation



S-Box 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-Box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	45	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Abbildung 4: eigene Abbildungen nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003 ², S.61, 62

5.4 Die 8 S-Boxen

Die S-Boxen stellen den mit Abstand wichtigsten Teil der DES-Verschlüsselung dar. Sie sind verantwortlich für die Sicherheit des zu verschlüsselnden Textes. Die S-Boxen sind konstant im Algorithmus eingebaut und werden, wie vorher beschrieben angewendet. Zum Beispiel kann man annehmen der 6Bit-Block für die S-Box 3 ist binär '100110'. Dann wird für die Zeile das erste und letzte Bit zusammengefügt ('10') und als Zahl interpretiert (2). Also ist die auszugebende Zahl in der zweiten Zeile zu finden, wobei man bei Null anfängt zu zählen. Die Bits 2 bis 5 binär zusammengefasst ergeben '0011'. Dies als Zahl interpretiert ergibt den Wert 3. Also findet sich unsere gesuchte Zahl in Zeile 2 und Spalte 3 (wieder wird von Null angefangen zu zählen). Unsere Ausgabe der S-Box lautet also '9' (siehe Abbildung 4 „S-Box 3“). Dies als 4Bit-Zahl ausgegeben ist '1001'.

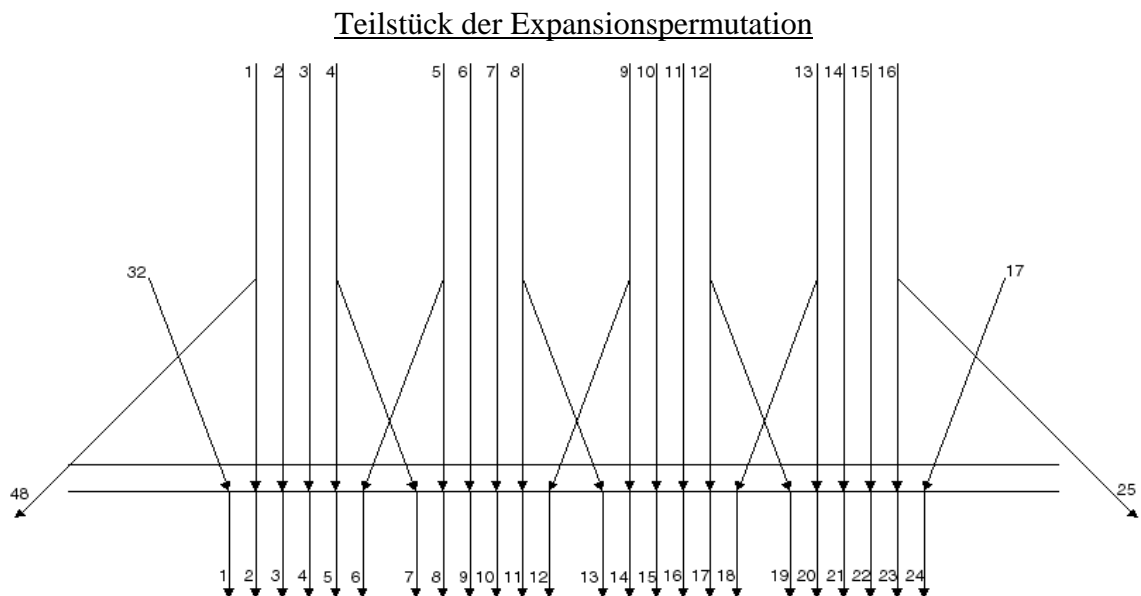


Abbildung 5: Abbildung von <http://members.chello.at/s.peer/DES/> am 8 März '05

5.5 Die Expansionspermutation

Die Expansionspermutation, auch E-Box genannt, erweitert einen 32Bit-Block auf einen 48Bit-Block. Dies geschieht, indem die Expansionspermutation jedes vierte und das darauf folgende Bit doppelt abbildet. Zum Beispiel bildet die Expansionspermutation das ursprünglich vierte Bit auf die Stelle 5 und 7 ab, und das ursprüngliche fünfte Bit wird an Stelle 6 und 8 abgebildet (siehe Abbildung 5 „Teilstück der Expansionspermutation“).

P-Box-Permutation

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

Abbildung 6: eigene Abbildung nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003 ², S.59

5.6 Die P-Box-Permutation

Die P-Box Permutation ist eine einfache Vertauschung der ursprünglichen Reihenfolge der 32Bit-Blöcke. Sie ignoriert kein Bit der ursprünglichen Reihenfolge und vervielfacht auch keins. Die P-Box ist konstant und somit bei jedem DES-Verschlüsselungsverfahren gleich. Zum Beispiel wird Bit 16 auf Bitposition 1 abgebildet, Bit 7 auf Bitposition 2... (siehe Abbildung 6 „P-Box-Permutation“).

Schlüsselverschiebung

Runde	1	2	3	4	5	6	7	8
Anzahl	1	1	2	2	2	2	2	2
Runde	9	10	11	12	13	14	15	16
Anzahl	1	2	2	2	2	2	2	1

Schlüsselpermutation

57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Abbildung 7: eigene Abbildungen nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003², S.59

5.7 Die Rundenschlüssel

Der als Eingabe noch 64Bit entsprechende Schlüssel für das DES-Verfahren wird durch die Schlüsselpermutation auf 56Bit verkürzt. Dies geschieht, indem jedes achte Bit ignoriert wird und die verbleibenden 56Bit neu angeordnet werden, was wiederum bedeutet, dass der effektive Schlüsselraum aus 56Bit besteht (Abschnitt 7.2).² Da jedes achte Bit ein sog. Paritätsbit (=Prüfbit) ist, besteht kein Informationsverlust. Nach dem der Schlüssel verkürzt worden ist, wird er in zwei Hälften, A_0 und B_0 , zu je 28Bit geteilt. In der Runde i werden nun die beiden Hälften A_i und B_i , abhängig von ihrem Rundenindex, entsprechend der Schlüsselverschiebung um ein oder zwei Stellen nach links verschoben, und dienen in der nächsten Runde als neue Hälften A_{i+1} und B_{i+1} . Die Hälften A_i und B_i werden in jeder neuen Runde zusammengefügt und durch die Kompressionspermutation (Abschnitt 5.8) 48 der 56Bit als Rundenschlüssel an die vorher beschriebene Funktion f übergeben.

² Q6, 9.3. '05

Kompressionspermutation

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Abbildung 8: eigene Abbildung nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003 ², S.59

5.8 Die Kompressionspermutation

Die Kompressionspermutation, ist eine konstante S-Box des DES-Verfahrens und ermittelt den aktuellen Rundenschlüssel für die Funktion f , indem sie die Bits des Schlüssels neu anordnet. Durch die zyklische Verschiebung der Teile A_i und B_i wird keine der 56Bit ausgelassen. Jedes Bit wird in den insgesamt 16 Runden als Rundenschlüssel verwendet.

8. Beispiel

Als erstes wird bei einer Verschlüsselung der Schlüssel eingegeben. Dieser besteht aus einem 16stelligen Hexadezimal Wert.

Schlüssel Hexadezimal:
F3CF94C2386A60A4

Dieser Schlüssel wird dann vom Programm in einen 64stelligen Binär Wert umgewandelt.

Schlüssel Binär (64Bit):
1111001111001111100101001100001000111000011010100110000010100100

Nachdem er in den Binär Wert umgewandelt worden ist, folgt die Schlüsselpermutation (Abbildung 7 „Schlüsselpermutation“).

Wert nach der Schlüsselpermutation (56Bit):
10001111011010111111000100010010101110000110001100100101

Nun wird vom Programm der Schlüssel in zwei hälften geteilt und um eine bestimmte Stellenanzahl (Abbildung 7 „Schlüsselverschiebung“) nach links verschoben (hier Runde 1).

Teilschlüssel 1 (28Bit):
0001111011010111111000100011

Teilschlüssel2 (28Bit):
0101011100001100011001001010

Nachdem der Schlüssel erfolgreich verschoben worden ist, wird er wieder zusammengefügt.

Teilschlüssel kombiniert (56Bit):
00011110110101111110001000110101011100001100011001001010

Der wieder zusammengefügte Teilschlüssel wird dann mit Hilfe der Kompressionspermutation (Abbildung 8 „Kompressionspermutation“) auf 48Bit verkürzt und zum Rundenschlüssel (hier Runde 1):

Rundenschlüssel nach der Kompressionspermutation (48Bit):

11000101110111110011100010001110000000011111101

Die anderen Rundenschlüssel der Verschlüsselung ergeben sich aus weiterem verschieben des 56Bit langem Teilschlüssels und weiterem übergeben an die Kompressionspermutation.

Nun folgt die Eingabe des Klartextes. Dieser wird im Programm gleichzeitig mit dem Schlüssel eingegeben.

Klartext:

Beispiel

Nachdem der Klartext eingegeben worden ist, wird dieser in seinen Binärwert zerlegt und jeweils als 64Bit Block der Verschlüsselung übergeben. Da der Klartext in diesem Fall nur aus 64Bit besteht, wird nur ein Block der Verschlüsselung übergeben.

Klartext als Binärwert (64Bit):

0100001001100101011010010111001101110000011010010110010101101100

Der Binärwert wird dann in der ersten Runde mit der Eingangspemutation (Abbildung 2 „Eingangspemutation“), und nach der letzten Runde mit der Ausgangspemutation (invers zur Eingangspemutation) neu angeordnet.

Klartext nach der Eingangspemutation (64Bit):

000000001111111100000000010000100000000111111110000000000000000

Nachdem die Eingangspemutation erfolgreich abgeschlossen wurde, teilt das Programm den Klartext in zwei hälften (Abbildung 1 „Die 16 Runden von DES“).

L0 (32Bit):

0000000011111111000000001000010

R0 (32Bit):

00000000111111110000000000000000

Nun folgt die Expansionspermutation (Abbildung 5 „Teilstück der Expansionspermutation“), die nur den R0 Teil betrifft und ihn auf 48Bit vergrößert.

R0 nach der Expansionspermutation (48Bit):

000000000010111111110100000000000000000000

Jetzt wird vom Programm der vergrößerte R0 Teil XOR mit dem aktuellen Rundenschlüssel verknüpft.

R0 nach der XOR Verknüpfung (48Bit):

11000101110010001100011000001110000000011111101

Daraufhin folgt der für die Sicherheit wichtigste Teil der Verschlüsselung, die S-Box-Transformation (Abbildung 4 „Die S-Box Transformation“).

R0 nach der S-Box-Transformation (32Bit):

01010101101000111011100100000110

Nachdem die S-Box-Transformation vom Programm durchgeführt wurde, folgt die P-Box-Permutation (Abbildung 6 „P-Box-Permutation“).

R0 nach der P-Box-Permutation (32Bit):

10110001010000101110000110110110

Nun wird der R0 Teil XOR mit dem L0 Teil verknüpft und zum R1 Teil (Abbildung 1 „Die 16 Runden von DES“).

R1 nach der XOR Verknüpfung von R0 und L0 (32Bit):

10110001101111011110000111110100

Der ursprüngliche R0 Teil wird zum neuen L1 Teil (Abbildung 1 „Die 16 Runden von DES“).

L1 (32Bit):

00000000111111110000000000000000

Nun ist eine komplette Runde des DES-Verfahrens durchlaufen. Insgesamt wird dies 16mal im gesamten Verschlüsselungsablauf durchgeführt um den fertigen Chiffretext zu erhalten.

7. Anhang

7.1 Schlüsselaustausch mit symmetrischen Verschlüsselungsverfahren

Wollen zwei Personen (hier Alice und Bob) auf einer Leitung miteinander Kommunizieren ergibt sich häufig ein Sicherheitsproblem. Dieses Problem wird beim symmetrischen Verschlüsselungsverfahren durch eine dritte Person, dem Trustcenter, gelöst. Es setzt voraus, dass Alice sowie Bob einen sicheren Schlüssel mit dem Trustcenter vereinbart haben. Im Fall von Alice = K_a . Im Fall von Bob = K_b . Um eine sichere Kommunikation gewährleisten zu können muss Alice vorab beim Trustcenter einen sicheren „Schlüssel“ für beide Kommunikationspartner anfordern.

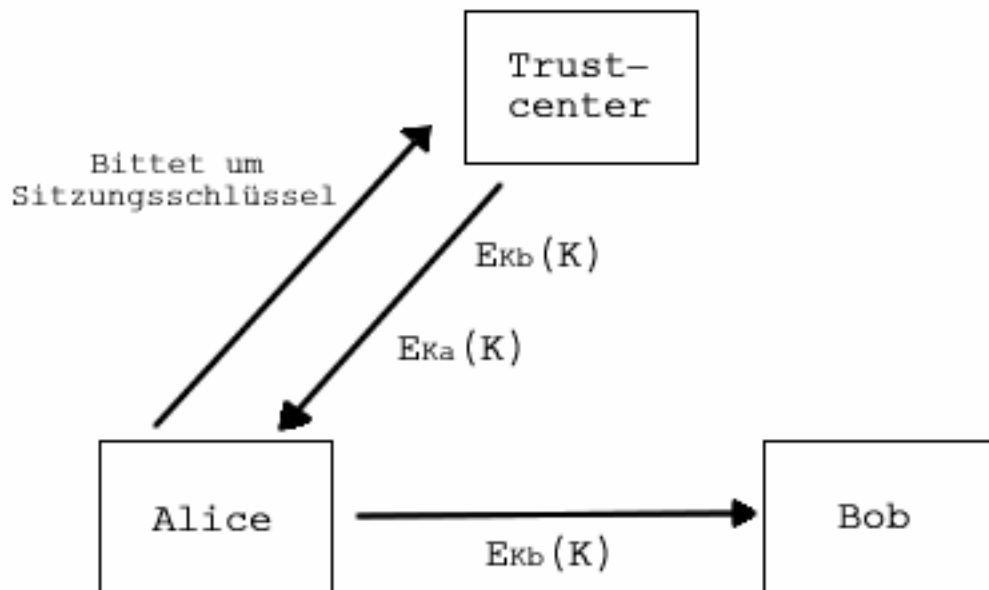


Abbildung 9: eigene Abbildung nach

Ertel Wolfgang: Angewandte Kryptographie, München 2003², S.88

Das Problem ist aber dennoch nicht ganz behoben. Ein weiterer kritischer Punkt ergibt sich beim Schlüsselaustausch durch das Trustcenter. Denn diese Vermittlungsstellen unterliegen selten Kontrollen und dennoch bekommen sie diskrete Daten zugeschickt um sie weiterzuleiten. Zum anderen wird bei einem eventuellen Angriff auf das Trustcenter das Netz unsicher, vielleicht sogar ohne dass die Teilnehmer etwas davon merken.

7.2 Die Länge des Schlüsselraums und die heutige Rechenleistung

Software-Implementierungen chiffrieren bei der heutigen Rechenleistung über eine Millionen DES-Blöcke pro Sekunde. Bei einem effektiven Schlüsselraum von $2^{56} \approx 7,2 \cdot 10^{16}$ würde bei einem Brute-Force-Angriff (stupides ausprobieren aller Schlüssel) ein Rechner mit dieser Leistungsstärke ca. 2284 Jahre benötigen um alle Schlüssel auszuprobieren. Geht man davon aus, dass der Schlüssel nach der Hälfte des zu versuchendem Schlüsselraums gefunden wird, würde ein derartiger Rechner immer noch ca. 1142 Jahre brauchen, um die Schlüsselanzahl durchzuprobieren.³ Im Rahmen der RSA-DES-Challenge-III wurde der bislang schnellste Brute-Force-Angriff gegen DES durchgeführt. 100.000 vernetzte Rechner und ein Spezialrechner knackten einen vorgegebenen 88 Byte langen Chiffretext in 22 Stunden. Sie erreichten zusammen eine Rechenleistung von $2,45 \cdot 10^{11}$ DES-Blöcken pro Sekunde. Dieser Angriff war erfolgreich nach etwas mehr als einem Viertel des Schlüsselraums. Zum durchsuchen des gesamten Schlüsselraums hätten diese Rechner, trotz ihrer hohen Rechenleistung, ca. 82 Stunden benötigt.

³ Q1, 64

8. Literaturverzeichnis

- Q1: Ertel, Wolfgang: Angewandte Kryptographie
Carl Hanser Verlag München Wien 2003 ²
- Q2: Kryptologie – DES
<http://www.regenechsen.de/krypto/des.php>
- Q3: Department of ST&IT
<http://www.cast.uni-linz.ac.at/Courses/CoursesWS0405/Krypto/des.pdf>
- Q4: Gymnasium und Oberstufenrealgymnasium Sankt Johann in Tirol
<http://www.bg-stjohann.asn-ibk.ac.at/faecher/informatik/fischer/node40.html>
- Q5: Universität Osnabrück - Institut für Informatik
<http://www-lehre.inf.uos.de/~rspier/referat/internet/DES-Algorithmus.html>
<http://www-lehre.inf.uos.de/~rspier/referat/internet/substibox.html>
- Q6: Linux Magazin
<http://www.linux-magazin.de/Artikel/ausgabe/1997/09/Krypto/krypto3.html>