Die S-Box von Byte Sub:

99	124	119	123	242	107	111	197
48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240
173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204
52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154
7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160
82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91
106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133
69	249	2	127	80	60	159	168
81	163	64	143	146	157	56	245
188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23
196	167	126	61	100	93	25	115
96	129	79	220	34	42	144	136
70	238	184	20	222	94	11	219
22[illegible]	50	58	10	73	6	36	92
194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169
108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198
232	221	116	31	75	189	139	138
112	62	181	102	72	3	246	14
97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148
155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104
65	153	45	15	176	84	187	22

Shift Row

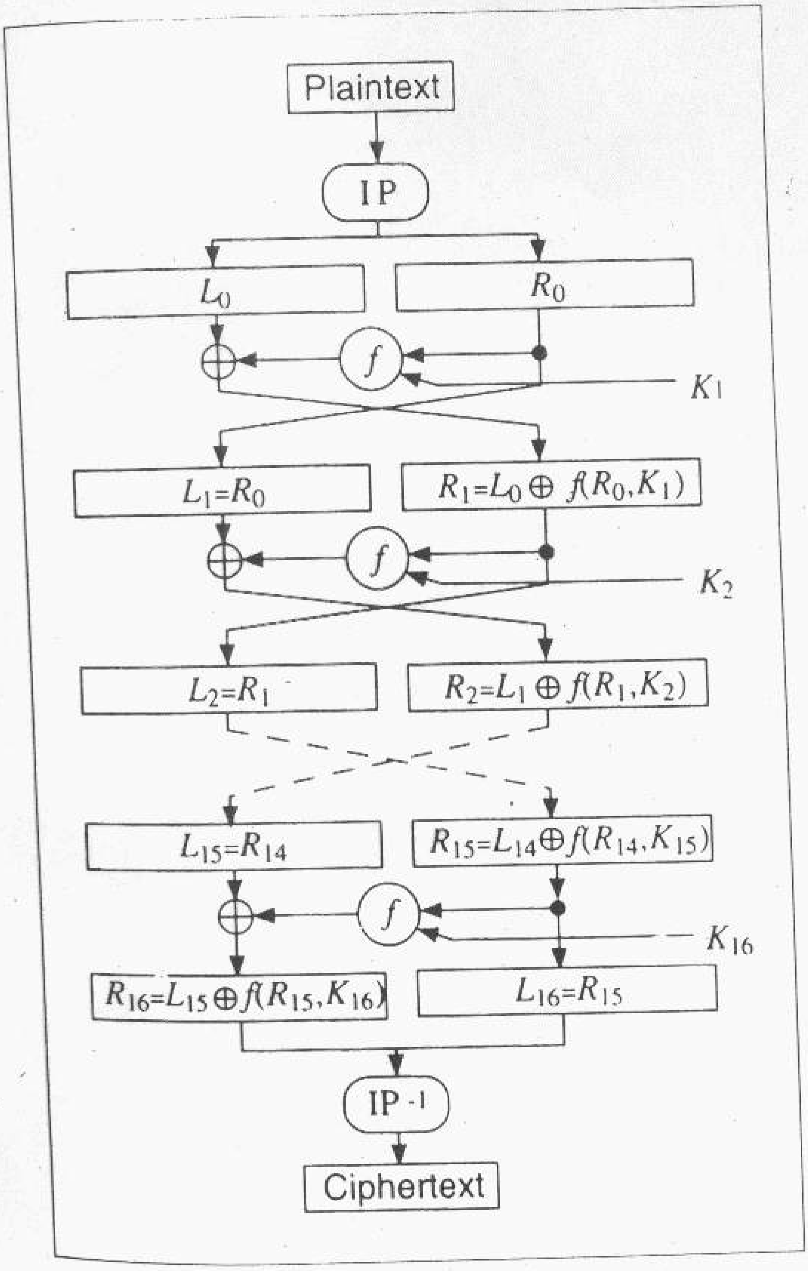
from				to			
1	5	9	13	1	5	9	13
2	6	10	14	6	10	14	2
3	7	11	15	11	15	3	7
[illegible]	8	12	16	16	4	8	12

Mix Column Matrix

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Multiplikation von 11001010 mit 3 in GF(2^8):

```
   11001010
 *       11
  ---------
  11001010
   11001010
  ---------
  101011110   (XOR instead of addition)
  100011011   (this is XORed, instead of subtracting 256)
  ---------
    1000101
```



$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f\,(R_{i-1}, K_i)$$

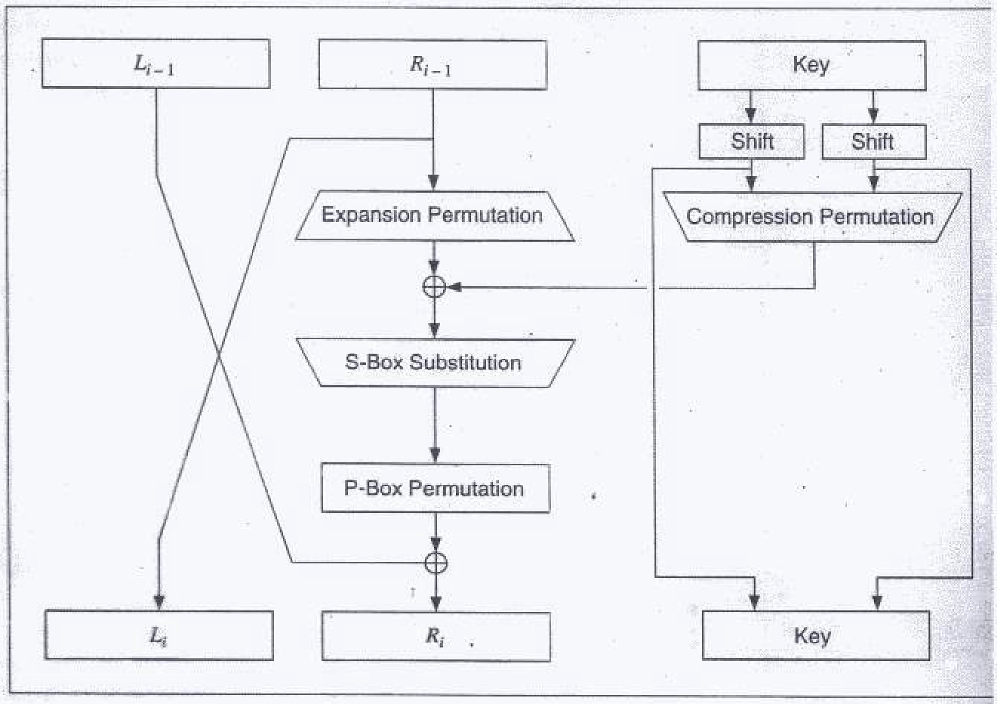


Figure 12.2 One round of DES.

Table 12.5
Expansion Permutation

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

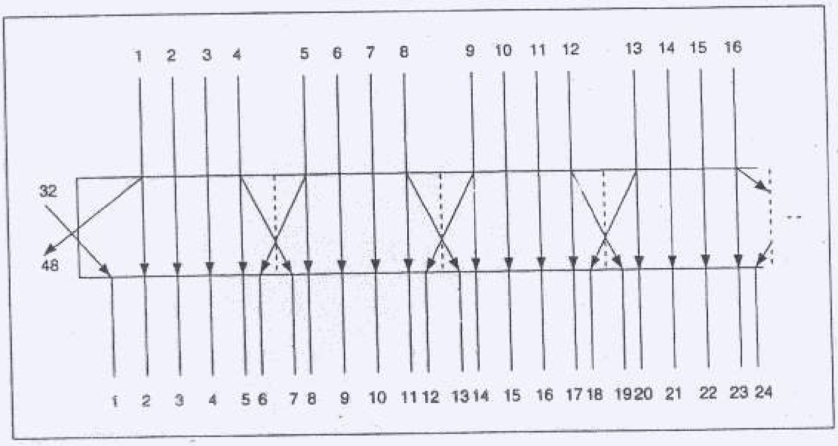


Figure 12.3 Expansion permutation.

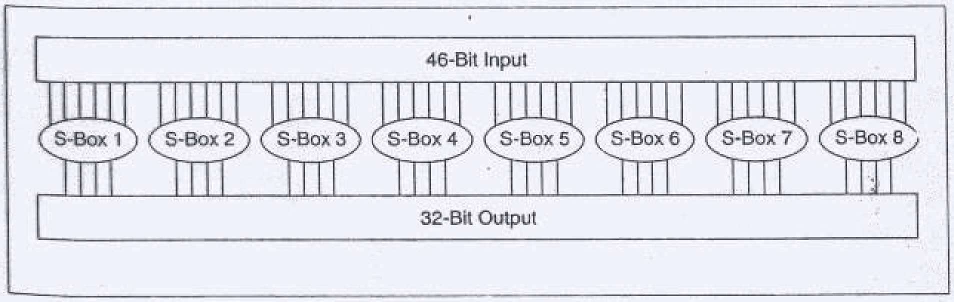


Figure 12.4 S-box substitution.

Table 12.7
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

Table 12.8
Final Permutation

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

Table 12.6
S-Boxes

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,	5,	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

Table 12.1
Initial Permutation

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

Table 12.2
Key Permutation

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

Table 12.3
Number of Key Bits Shifted per Round

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

Table 12.4
Compression Permutation

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

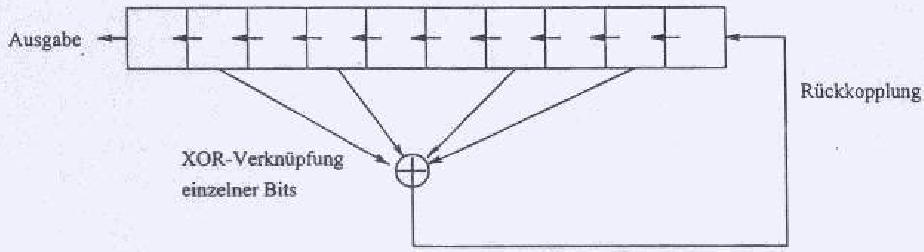


Abbildung 5.18: Ein 10 Bit langes Schieberegister mit linearer Rückkopplung (LFSR)

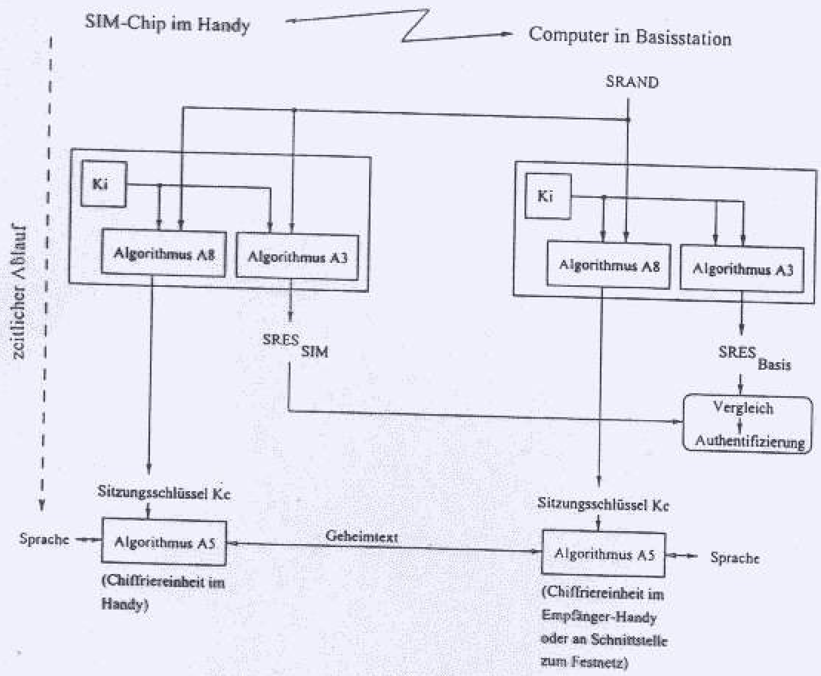


Abbildung 6.1: Authentifizierung und Erzeugung der Sitzungsschlüssel in GSM-Netzen

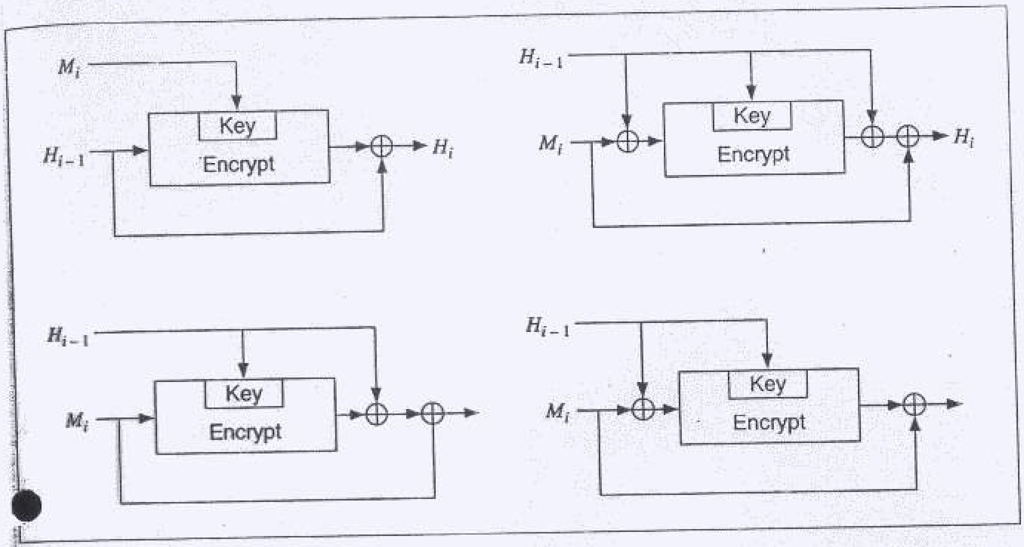


Figure 18.9 *The four secure hash functions where the block length equals the hash size.*

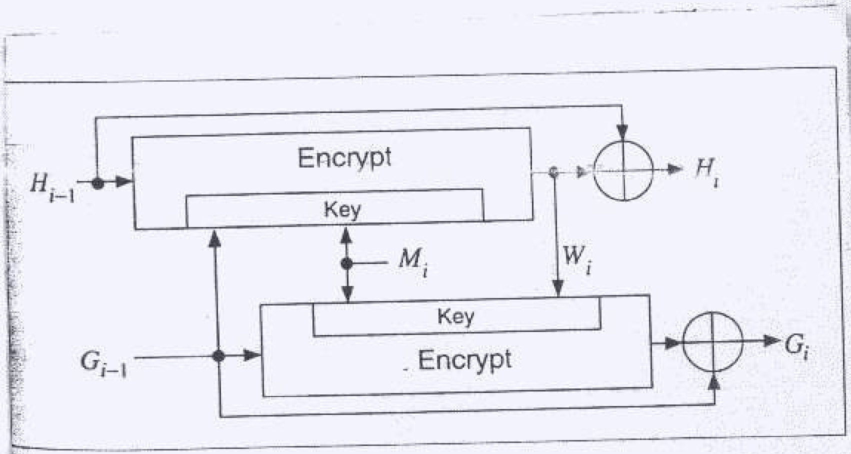


Figure 18.11 *Tandem Davies-Meyer.*

Table 19.4
RSA Speeds for Different Modulus Lengths with an 8-bit Public Key (on a SPARC II)

	512 bits	768 bits	1,024 bits
Encrypt	0.03 sec	0.05 sec	0.08 sec
Decrypt	0.16 sec	0.48 sec	0.93 sec
Sign	0.16 sec	0.52 sec	0.97 sec
Verify	0.02 sec	0.07 sec	0.08 sec

any of these three values for e (assuming you pad messages with random values—see later section), even if a whole group of users uses the same value for e.

Private key operations can be speeded up with the Chinese remainder theorem if you save the values of p and q, and additional values such as $d \bmod (p - 1)$, $d \bmod (q - 1)$, and $q^{-1} \bmod p$ [1283,1276]. These additional numbers can easily be calculated from the private and public keys.

Security of RSA

The security of RSA depends wholly on the problem of factoring large numbers. Technically, that's a lie. It is *conjectured* that the security of RSA depends on the problem of factoring large numbers. It has never been mathematically proven that you need to factor n to calculate m from c and e. It is conceivable that an entirely different way to cryptanalyze RSA might be discovered. However, if this new way allows the cryptanalyst to deduce d, it could also be used as a new way to factor large numbers. I wouldn't worry about it too much.

It is also possible to attack RSA by guessing the value of $(p - 1)(q - 1)$. This attack is no easier than factoring n [1616].

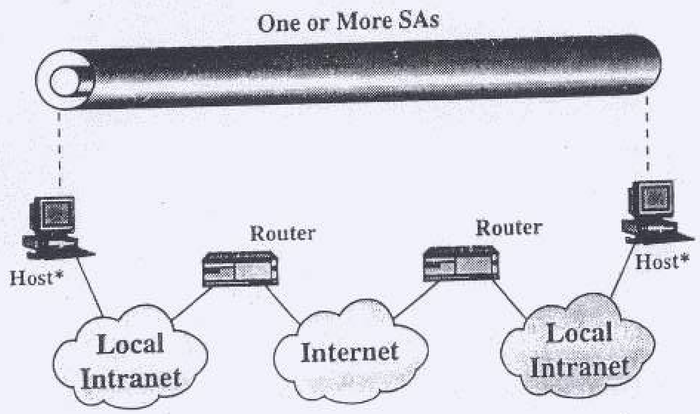
For the ultraskeptical, some RSA variants have been proved to be as difficult as factoring (see Section 19.5). Also look at [36], which shows that recovering even certain bits of information from an RSA-encrypted ciphertext is as hard as decrypting the entire message.

Factoring n is the most obvious means of attack. Any adversary will have the public key, e, and the modulus, n. To find the decryption key, d, he has to factor n. Section 11.4 discusses the current state of factoring technology. Currently, a 129-decimal-digit modulus is at the edge of factoring technology. So, n must be larger than that. Read Section 7.2 on public key length.

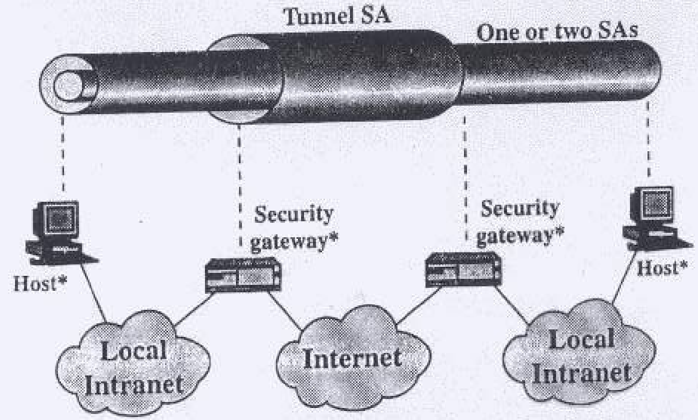
It is certainly possible for a cryptanalyst to try every possible d until he stumbles on the correct one. This brute-force attack is even less efficient than trying to factor n.

From time to time, people claim to have found easy ways to break RSA, but to date no such claim has held up. For example, in 1993 a draft paper by William Payne proposed a method based on Fermat's little theorem [1234]. Unfortunately, this method is also slower than factoring the modulus.

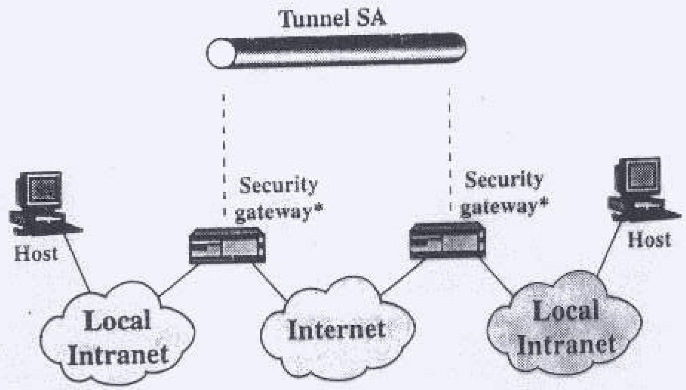
There's another worry. Most common algorithms for computing primes p and q are probabilistic; what happens if p or q composite? Well, first you can make the odds of that happening as small as you want. And if it does happen, the odds are that



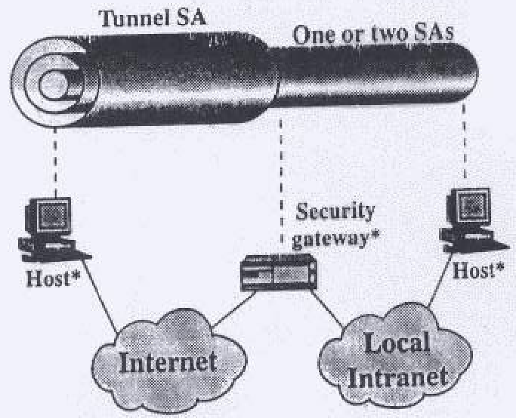
(a) Case 1



(c) Case 3



(b) Case 2



(d) Case 4

* = implements IPSec

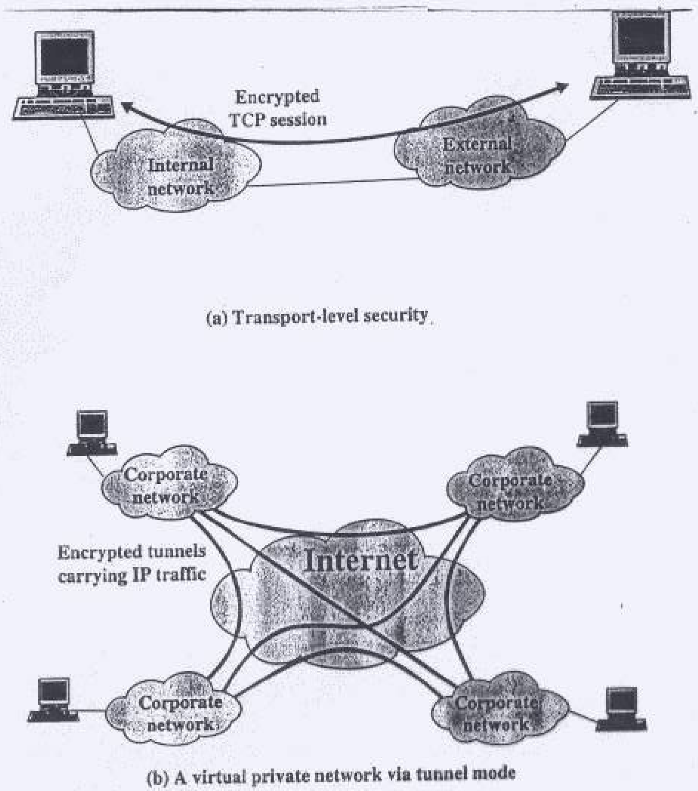


Figure 13.8 Transport Mode versus Tunnel Mode Encryption.

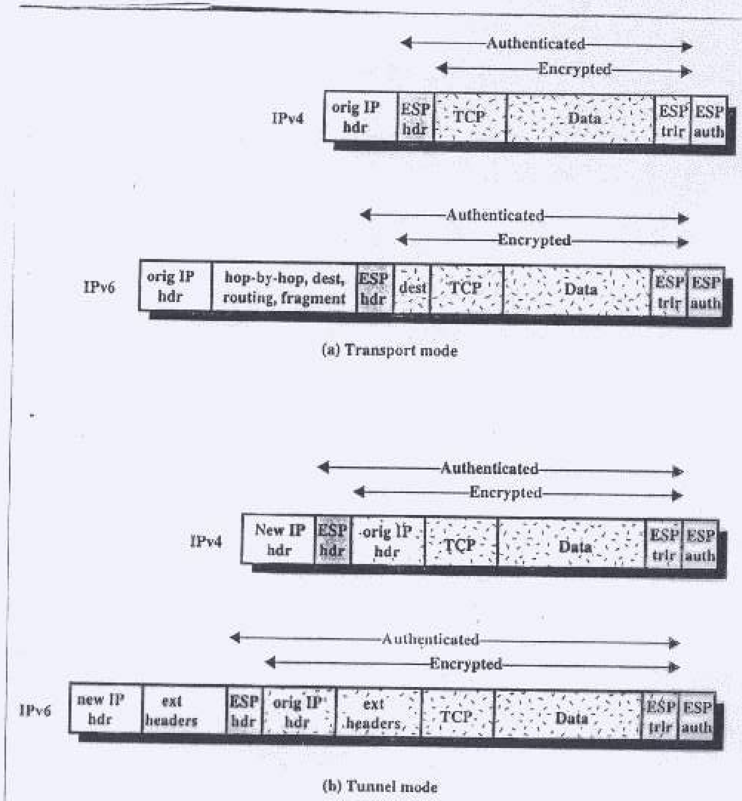


Figure 13.9 Scope of ESP Encryption and Authentication.

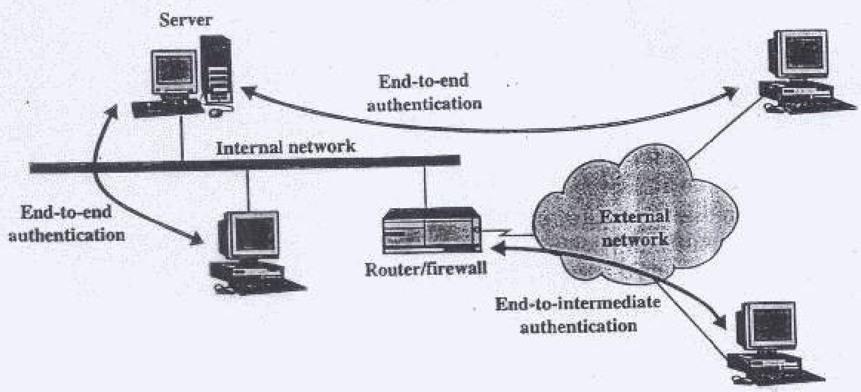
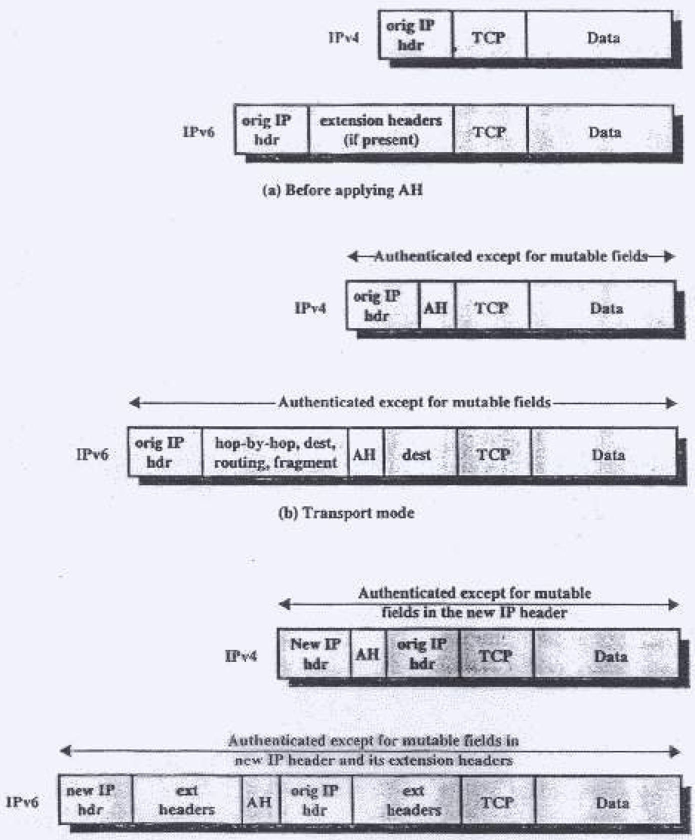


Figure 13.5 End-to-end versus End-to-intermediate Authentication.



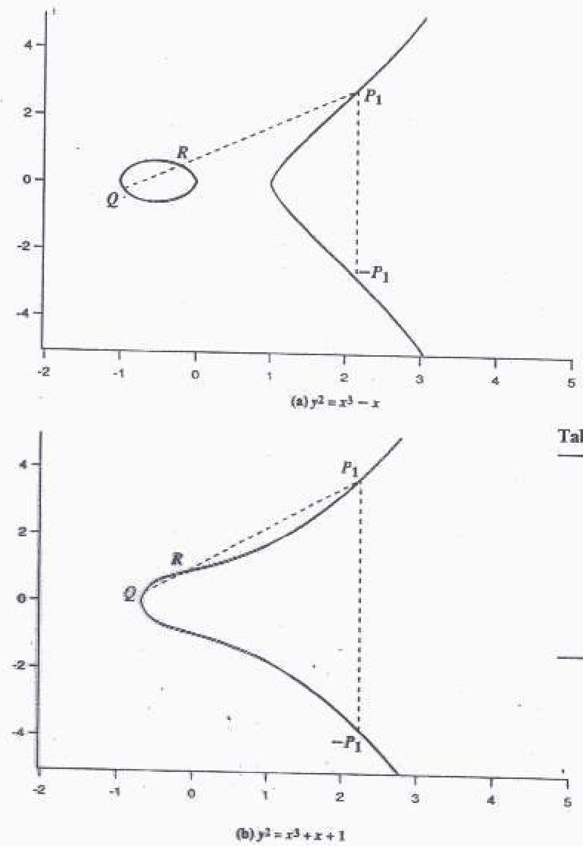


Figure 6.18 Example of Elliptic Curves.

Table 6.4 Points on the Elliptic Curve $E_{23}(1, 1)$

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

We look at two examples, taken from [JURI97]. Let $P = (3, 10)$ and $Q = (9, 7)$. Then

$$\lambda = \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} \equiv 11 \bmod 23$$
$$x_3 = 11^2 - 3 - 9 = 109 \equiv 17 \mod 23$$
$$y_3 = 11(3 - (-6)) - 10 = 89 \equiv 20 \mod 23$$

So $P + Q = (17, 20)$. To find $2P$,

$$\lambda = \frac{3(3^2) + 1}{2 \times 10} = \frac{5}{20} = \frac{1}{4} \equiv 6 \bmod 23$$
$$x_3 = 6^2 - 3 - 3 = 30 \equiv 7 \mod 23$$
$$y_3 = 6(3 - 7) - 10 = -34 \equiv 12 \mod 23$$

and $2P = (7, 12)$.

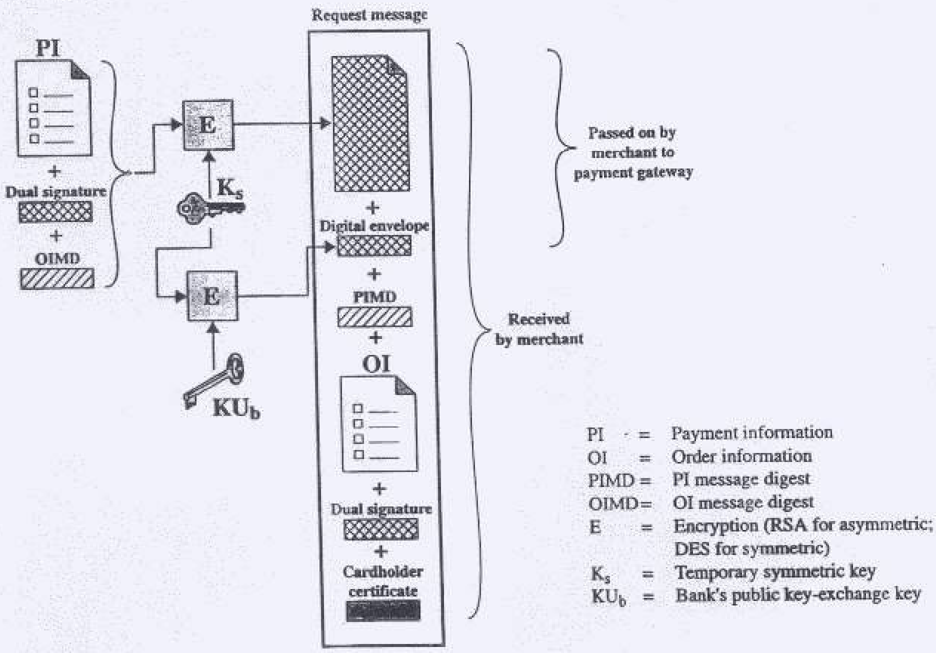


Figure 14.10 Cardholder Sends Purchase Request.

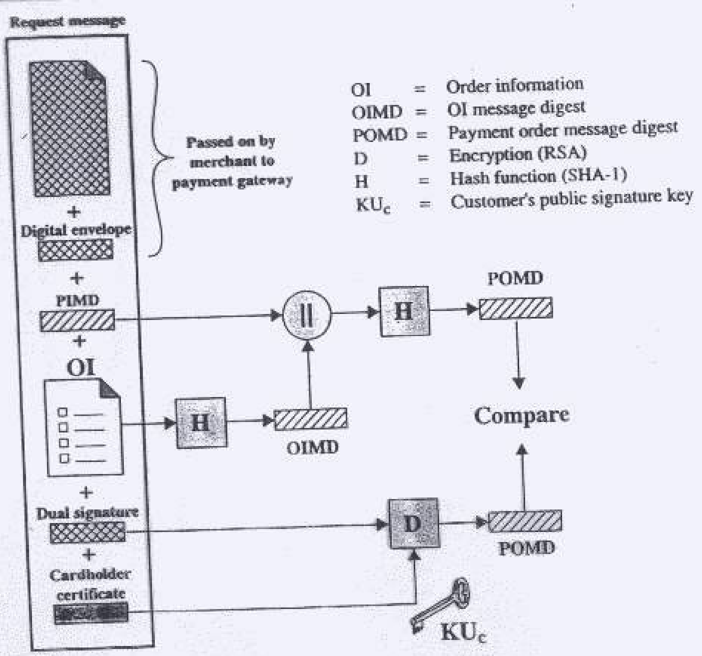


Figure 14.11 Merchant Verifies Customer Purchase Request.

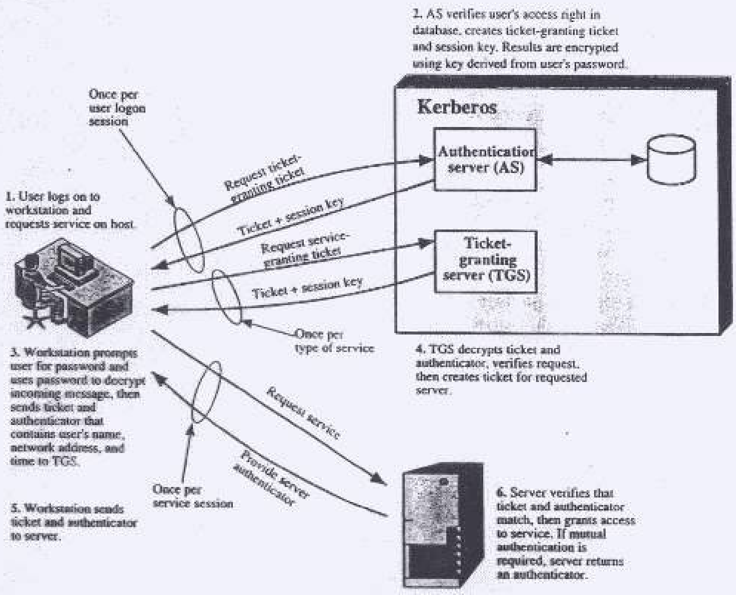


Figure 11.1 Overview of Kerberos.

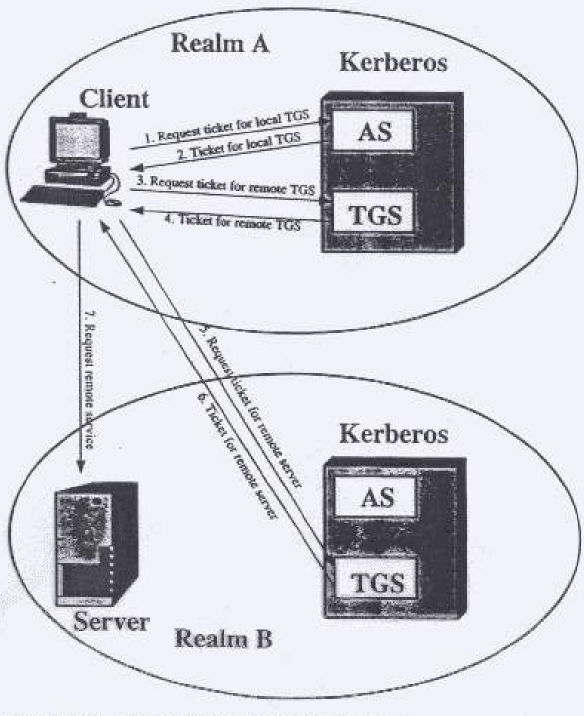


Figure 11.2 Request for Service in Another Realm.

(1) $C \rightarrow AS$: $ID_C||P_C||ID_V$

(2) $AS \rightarrow C$: Ticket

(3) $C \rightarrow V$: $ID_C||Ticket$

$Ticket = E_{K_V}[ID_C||AD_C||ID_V]$

C	=	client
AS	=	authentication server
V	=	server
ID_C	=	identifier of user on C
ID_V	=	identifier of V
P_C	=	password of user on C
AD_C	=	network address of C
K_V	=	secret encryption key shared by AS and V
$\|\|$	=	concatenation

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C||ID_{tgs}$

(2) $AS \rightarrow C$: $E_{K_C}[Ticket_{tgs}]$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C||ID_V||Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_V$

Once per service session:

(5) $C \rightarrow V$: $ID_C||Ticket_V$

$Ticket_{tgs} = E_{K_{tgs}}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V = E_{K_V}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

Table 11.1 Summary of Kerberos Version 4 Message Exchanges

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) **C → AS:** $ID_c \| ID_{tgs} \| TS_1$

(2) **AS → C:** $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2]$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) **C → TGS:** $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C:** $E_{K_{c,tgs}}[K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2]$

$Ticket_v = E_{K_v}[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4]$

$Authenticator_c = E_{K_{c,tgs}}[ID_c \| AD_c \| TS_3]$

(c) Client/Server Authentication Exchange: to obtain service

(5) **C → K:** $Ticket_v \| Authenticator_c$

(6) **K → C:** $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$Ticket_v = E_{K_v}[K_{c,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4]$

$Authenticator_c = E_{K_{c,v}}[ID_c \| AD_c \| TS_5]$

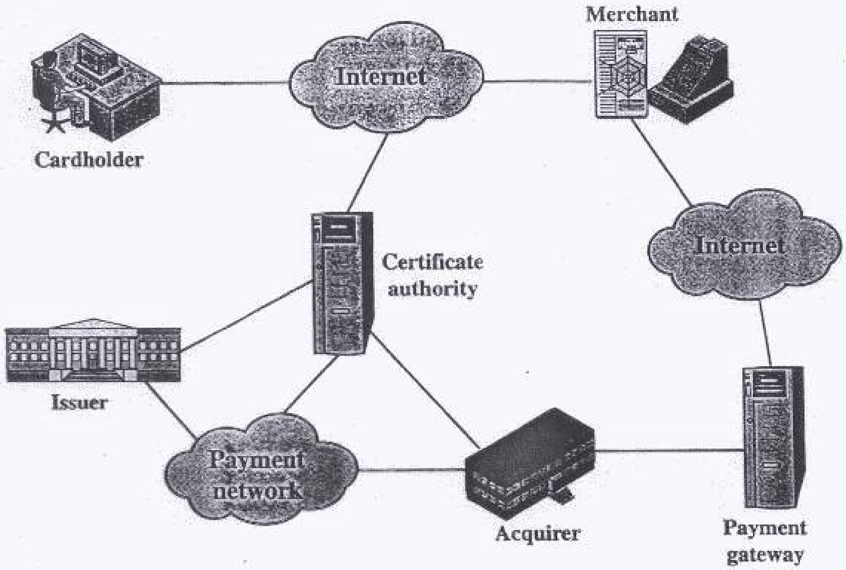


Figure 14.8 Secure Electronic Commerce Components.

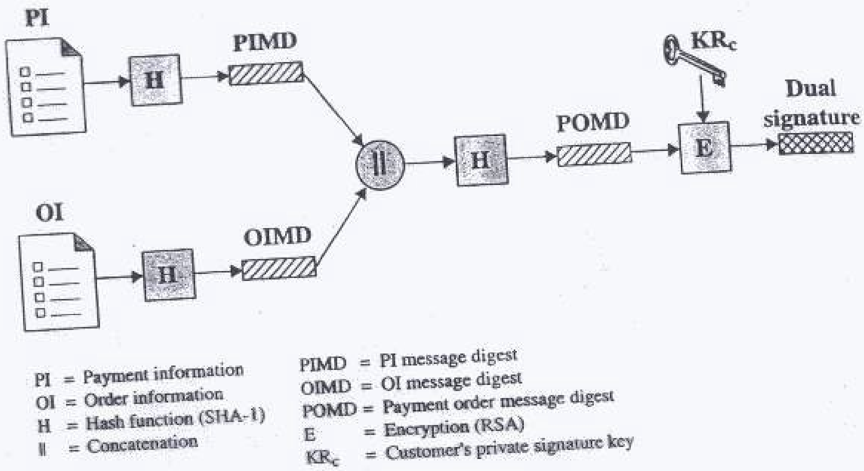


Figure 14.9 Construction of Dual Signature.

Table 11.2 Rationale for the Elements of the Kerberos Version 4 Protocol

(a) Authentication Service Exchange

Message (1)	Client requests ticket-granting ticket
ID_C:	Tells AS identity of user from this client
ID_{tgs}:	Tells AS that user requests access to TGS
TS_1:	Allows AS to verify that client's clock is synchronized with that of AS
Message (2)	AS returns ticket-granting ticket
E_{K_c}:	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2)
$K_{c,tgs}$:	Copy of session key accessible to client; created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key
ID_{tgs}:	Confirms that this ticket is for the TGS
TS_2:	Informs client of time this ticket was issued
$Lifetime_2$:	Informs client of the lifetime of this ticket
$Ticket_{tgs}$:	Ticket to be used by client to access TGS

continued

(b) Ticket-Granting Service Exchange

Message (3)	Client requests service-granting ticket
ID_V:	Tells TGS that user requests access to server V
$Ticket_{tgs}$:	Assures TGS that this user has been authenticated by AS
$Authenticator_c$:	Generated by client to validate ticket
Message (4)	TGS returns service-granting ticket
$E_{K_{c,tgs}}$:	Key shared only by C and TGS; protects contents of message (4)
$K_{c,tgs}$:	Copy of session key accessible to client; created by TGS to permit secure exchange between client and server without requiring them to share a permanent key
ID_V:	Confirms that this ticket is for server V
TS_4:	Informs client of time this ticket was issued
$Ticket_V$:	Ticket to be used by client to access server V
$Ticket_{tgs}$	Reusable so that user does not have to reenter password
$E_{K_{tgs}}$:	Ticket is encrypted with key known only to AS and TGS, to prevent tampering
$K_{c,tgs}$:	Copy of session key accessible to TGS; used to decrypt authenticator, thereby authenticating ticket
ID_C:	Indicates the rightful owner of this ticket
AD_C:	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_{tgs}:	Assures server that it has decrypted ticket properly
TS_2:	Informs TGS of time this ticket was issued
$Lifetime_2$:	Prevents replay after ticket has expired
$Authenticator_c$:	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$E_{K_{c,tgs}}$:	Authenticator is encrypted with key known only to client and TGS, to prevent tampering
ID_C:	Must match ID in ticket to authenticate ticket
AD_C:	Must match address in ticket to authenticate ticket
TS_3:	Informs TGS of time this authenticator was generated

(c) Client/Server Authentication Exchange

Message (5)	Client requests service
$Ticket_V$:	Assures server that this user has been authenticated by AS
$Authenticator_c$:	Generated by client to validate ticket
Message (6)	Optional authentication of server to client
$E_{K_{c,v}}$:	Assures C that this message is from V
$TS_5 + 1$:	Assures C that this is not a replay of an old reply
$Ticket_v$	Reusable so that client does not need to request a new ticket from TGS for each access to the same server
E_{K_v}:	Ticket is encrypted with key known only to TGS and server, to prevent tampering
$K_{c,v}$:	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket
ID_C:	Indicates the rightful owner of this ticket
AD_C:	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_V:	Assures server that it has decrypted ticket properly
TS_4:	Informs server of time this ticket was issued
$Lifetime_4$:	Prevents replay after ticket has expired
$Authenticator_c$:	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$E_{K_{c,v}}$:	Authenticator is encrypted with key known only to client and server, to prevent tampering
ID_C:	Must match ID in ticket to authenticate ticket
AD_C:	Must match address in ticket to authenticate ticket
TS_5:	Informs server of time this authenticator was generated