Part XII

From theory to practice in cryptography

FROM CRYPTO-THEORY to CRYPTO-PRACTICE

In this chapter we deal with several applied cryptography methods, systems and problems that have played very important role in applications.
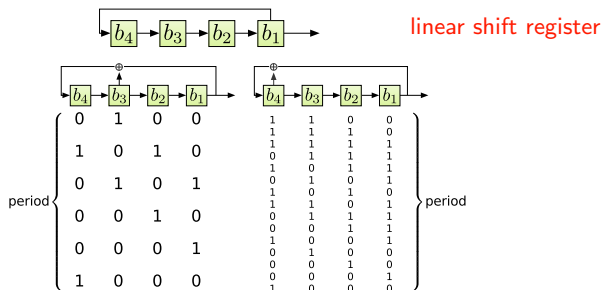
I.SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

Basic idea: to use a short key, called "seed", and a pseudorandom generator to generate long pseudorandom key.
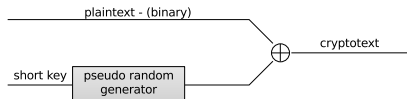
Shift registers as pseudorandom generators

linear shift register

$$\text{period} \begin{cases} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{cases}$$
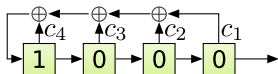
Theorem For every $n > 0$ there is a linear shift register of maximal period $2^n - 1$.

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of plaintext and of cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110
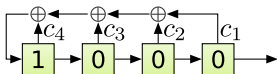
We need to determine c_4, c_3, c_2, c_1 such that the above sequence is outputted by the shift register

states of cell 4	states of cell 3	states of cell 2	states of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

$$c_4 = 1 \qquad c_4 = 1$$
$$c_4 \oplus c_3 = 1 \qquad c_3 = 0$$
$$c_2 \oplus c_4 = 1 \qquad \Longrightarrow \qquad c_2 = 0$$
$$c_1 \oplus c_3 \oplus c_4 \oplus c_3 \cdot c_4 = 0 \qquad c_1 = 1$$

COMPUTATIONS

states of cell 4	states of cell 3	states of cell 2	states of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

After the second step new value of the first register is

$$N = (c_4 \cdot c_4) \oplus c_3 = c_4 \oplus c_3$$

After the third step new value of the first register is

$$N = ((c_4 \oplus c_3) \cdot c_4) \oplus (c_4 \cdot c_3) \oplus c_2$$

If $c_4 = 1$, then

$$N = \bar{c_3} \oplus c_3 \oplus c_2 = \bar{c_2}$$

and therefore $N = c_4 \oplus c_2$.
If $c_4 = 0$, then

$$N = c_2$$

and therefore $N = c_4 \oplus c_2$

Linear Recurrences

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \pmod{n}$$

that can be specified by $2m$ bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

The main advantage of such recurrences is that a pseudo-random key of a very large period can be generated using a few initial bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

Encryption using one-time pad and a key generated by a linear feedback shift register succumbs easily to a known plaintext attack. If we know few bits of the plaintext and of the corresponding cryptotext, one can easily determine the initial part of the key and then the corresponding linear recurrence, as already shown.

To test whether a given portion of a key was generated by a recurrence of a length m, if we know $x_1, \ldots, x_{2m}$, we need to solve the matrix equation

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_m \\ x_2 & x_3 & \ldots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \ldots & x_{2m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} x_{m+1} \\ x_{m+2} \\ \vdots \\ x_{2m} \end{pmatrix}$$

and then to verify whether the remaining available bits, $x_{2m+1}, \ldots$, are really generated by the recurrence obtained.

FINDING LINEAR RECURRENCES

The basic idea to find linear recurrences generating a given sequence is to check whether there is such a recurrence for m = 2, 3, ... In doing that we use the following result.

Theorem Let

$$M = \begin{pmatrix} x_1 & x_2 & \ldots & x_m \\ x_2 & x_3 & \ldots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \ldots & x_{2m-1} \end{pmatrix}$$

If the sequence $x_1, x_2 \ldots, x_{2m-1}$ satisfies a linear recurrence of length less than m, then $\det(M) = 0$.

Conversely, if the sequence $x_1, x_2 \ldots, x_{2m-1}$ satisfies a linear recurrence of length m and $\det(M) = 0$, then the sequence also satisfies a linear recurrence of length less than m.

II. STREAM CRYPTOSYSTEMS

BLOCK versus STREAM CRYPTOSYSTEMS

Two basic types of cryptosystems are:

- **Block cryptosystems:** (Hill cryptosystem, ...) - they are used to encrypt blocks of plaintext, one by one, using a fixed encryption algorithm.
- **Stream cryptosystems:** (CAESAR, ONE-TIME PAD, ...) - they encrypt plaintext characters one by one using encryption transformations that may change with time.

Comments on stream cryptosystems:

- In some applications they are more appropriate and even mandatory (for example in telecommunications).
- Stream cryptosystems are usually simpler to implement in hardware and much faster.
- Stream cryptosystems have limited, or no, error propagation and therefore may be advantageous in situations where transmission errors are highly probable.

Two types of stream cryptosystems are:

- Secret key cryptosystems (ONE-TIME PAD).
- Public key cryptosystems (Blum-Goldwasser probabilistic cryptosystem).

SYNCHRONOUS STREAM CRYPTOSYSTEMS

In synchronous stream cryptosystems the key-stream is usually generated independently of the plaintext-stream and cryptotext-stream.

The encryption process can be described, for example, by equations:

$$\sigma_{i+1} = f(\sigma_i, k_i) \tag{1}$$

$$z_i = g(\sigma_i, k_i) \tag{2}$$

$$c_i = h(z_i, m_i) \tag{3}$$

where k_i is ith symbol of the key, σ_i are states with σ_0 as the initial state, f is the next state function, g is a function producing keystream elements z_i, h is cryptotext producing function. m_i is the ith symbol of the plaintext.

General model of a finite-automata model of a synchronous stream cryptosystem:

(a) Encryption   (b) Decryption

PROPERTIES of SYNCHRONOUS STREAM CRYPTOSYSTEMS
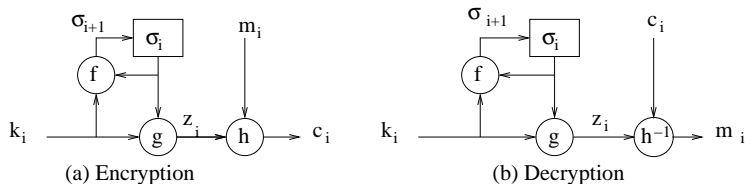
Properties of synchronous stream cryptosystems:

- Both the sender and the receiver have to be synchronized.
- There is no error propagation in case a cryptotext symbol is modified.
- An active attacker can cause a lot of damage. A use of data authentication techniques is necessary.

SELF-SYNCHRONIZING STREAM CRYPTOSYSTEMS
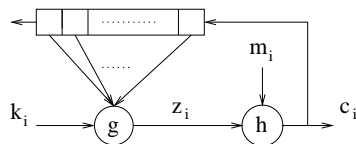
The encryption process:

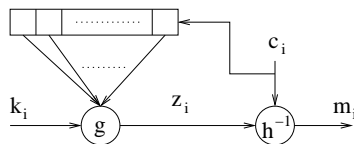$$\sigma_i = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}) \tag{4}$$

$$z_i = g(\sigma_i, k_i) \tag{5}$$

$$c_i = h(z_i . m_i) \tag{6}$$

can be depicted as follows

(a) Encryption   (b) Decryption

Properties of self-synchronizing cryptosystems

- Self-synchronization is possible if cryptotext digits are deleted/inserted.
- Limited error propagation.

RC4 – STREAM CRYPTOSYSTEM I

RC4 was designed by R. Rivest in 1987 and kept as a commercial secret till 1994. Some internet browsers/servers used RC4.

RC4 works as a finite automaton with an internal state. Its initial state is derived from the secret key only. Its internal state and next byte of the plaintext determine its next internal state and a new byte of the cryptotext, by making XOR of last bytes of plaintext and key..

The internal state consists of a triple (i, j, s), where i and j are bytes and s is a permutation on the set

$$\{0, 1, \ldots, 255\}$$

of bytes and it is encoded as an array $s[0], s[1], \ldots, s[255]$. Key is represented as an array

$$k[0], k[1], \ldots, k[l-1]$$

of $l \in [5, 16]$ bytes. The initial state is then designed as follows:

$j \leftarrow 0$;
for $i = 0$ **to** 255 **do** $S[i] \leftarrow i$;
for $i = 0$ **to** 225 **do**
$\qquad j \leftarrow j + s[i] + k[i \bmod l]$; swap $S[i]$ and$S[j]$
$i \leftarrow 0$; $j \leftarrow 0$

Plaintexts are iteratively encrypted and the initial state for a new plaintext is equal to the final state of the previous plaintext.

Key-stream generator:

$i \leftarrow i + 1; j \leftarrow j + S[i];$
swap $S[i]$ and $S[j]$;
output $S[S[i] + S[j]]$

A5/1 is based on a finite automaton that is based on the following three LFSRs (linear feedback shift registers) with a mutual shift control.
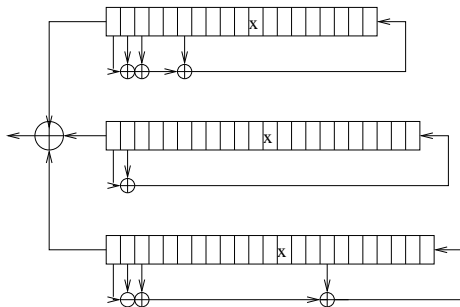
Three registers R_1, R_2 and R_3, contain $19 + 22 + 23 = 64$ bits. Every time unit some of the registers are shifted - that is their content is shifted by one position and one new bit is pushed in. The new bit is the XOR of a few bits of the three LFSRs involved.

A5/1 - STREAM CRYPTOSYSTEM - ENCRYPTION II

At each step those registers are shifted that have in a special cell, denoted by x, such a bit that is in the majority of bits of all three special cells.

Initiation phase (that uses a 64-bit secret key register K):

1: set all registers to zero;
2: for $i = 0$ to 63 do
$$R_1[0] \leftarrow R_1[0] \oplus K[i];$$
$$R_2[0] \leftarrow R_2[0] \oplus K[i];$$
$$R_3[0] \leftarrow R_3[0] \oplus K[i];$$
3. Shift all registers;
4. for $i = 0$ to 21 do
$$R_1[0] \leftarrow R_1[0] \oplus \text{count}[i];$$
$$R_2[0] \leftarrow R_2[0] \oplus \text{count}[i];$$
$$R_3[0] \leftarrow R_3[0] \oplus \text{count}[i];$$
shift all registers;
5. for $i = 0$ to 99 do shift the automaton

where "count" is a 22-bit registers that counts "frames" of the plaintexts, where each frame has 114 bites.

All that corresponds to 4 hours of GSM communication.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams.

Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.

Let

$$p_1, \ldots, p_n$$

be a plaintext.

Define for $0 \leq i < s$, $p_{-i} = k_{s-i}$ and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j} \right) \mod 26, \ 1 \leq i \leq n$$

Confusion makes the relation between the cryptotext and plaintext as complex as possible.

Example: polyalphabetic substitutions.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures the relationship between the plaintext and the cryptotext, which makes much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion. Polyalphabetic cryptosystems use only confusion. In permutation cryptosystems only diffusion step is used. DES essentially uses a sequence of confusion and diffusion steps.

IV. DES CRYPTOSYSTEM and its FOLLOWERS

During the years 1966-1972 a need grew up to develop an encryption standard so that very different users can easily communicate using encrypted messages.

The idea was that people should be able to communicate secretly using identical encryption and decryption machines.

On May 15, 1973 American National Bureau of Standards formally requested proposals for a standard encryption system.

The main candidate was the cryptosystem LUCIFER developed in IBM by Horst Feistel.

After 3 years of arguing of experts, a 56-bit key version of Lucifer was accepted (supposedly only for the next 5 years) as the standard called DES (Data Encryption Standard) on November 23, 1976.

15. 5. 1973 National Bureau of Standards published a solicitation for a new cryptosystem.

This led to the development of so far the most often used cryptosystem

Data Encryption Standard – DES

DES was developed at IBM, as a modification of an earlier cryptosystem called Lucifer.

17. 3. 1975 DES was published for the first time.

After long ad heated public discussion, DES was adopted as a standard on 15. 1. 1977.

DES used to be reviewed by NBS every 5 years.

DES was a revolutionary step in the secret-key cryptography history:

Both encryption and decryption algorithms were made public!!!!!!

DESALGORITHMS – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.

A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$. The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where each of the strings L_0 and R_0 has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

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

where f is a fixed+public and easy-to-implement function.

The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

DES cryptosystem – Data Encryption Standard – 1977

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where each of the strings L_0 and R_0 has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

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

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

Decryption $\phi_{64}(c) = L_{16} R_{16}$ is computed and then the recurrence

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

is used to get L_i, R_i i = 15,. . .,1,0, w = $\phi_{64}^{-1}(L_0, R_0)$.

A secret 56-bit key k_{56} is chosen.

Eight bits in (new) positions 8, 16,..., 64 are added to the key, to make each byte of odd parity.

This step is useful for error detections in key distribution and in storage.

56 bits of the key are now subject to the following fixed+public permutation ϕ_{56}:

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

The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$.

Blocks C_i, D_i for $i = 1, 2, \ldots, 16$ are now constructed from blocks C_{i-1}, D_{i-1} by one or two left shifts according the following table

ple:

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

shift means a rotation of the bits one place to the left; after one left

Using a fixed and publicly known order,

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

16 subkeys k_i, each of 48 bits, are then created, each k_i from blocks C_i, D_i

DES -ENCRYPTION DETAILS

A fixed+public initial permutation ϕ_{64}

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

is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where L_0 (R_0) has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are then designed using the recurrences:

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

where f is a fixed+public and easy-to-implement function to be described next. The cryptotext is now

$$^{-1}(\text{ } _{6} R_{16})$$

FUNCTION f

The function f produces from a 32-bit block R_{i-1} and a 48-bit subkey K_i a 32-bit block as follows:

At first, the 32-bit block is expanded into 48-bits according the following table:

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

After this expansion two 48-bits blocks are XOR-ed - bit by bit.

DES - ENCRYPTION - CONTINUATION

The resulting block of 48 bits is now divided into eight 6-bit blocks $B_1, B_2, \ldots, B_8$ and j-th of these eight 6-bit blocks is transformed into a 4-bit block using table S_j. The first two of them are:

$$S_1$$

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

$$S_2$$

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

Transformation is performed as follows. For a given 8-bit string, the first and last bit determine a number $x \in \{0, 1, 2, 3\}$ and the middle four bits a number y. The number in x-row and y-column is then written in binary.

Example: for the string 110010, we have $x = 2$, $y = 9$ and the resulting number defined by S_1 is 15. Therefore the resulting string is 1111

KEY ELEMENTS of DES

- A cryptosystem is called **linear** if each bit of cryptotext is a linear combination of bits of plaintext.

- For linear cryptosystems there is a powerful decryption method - so-called linear cryptanalysis.

- The only components of DES that are non-linear are S-boxes.

- Some of original requirements for S-boxes:
  - Each row of an S-box should include all possible output bit combinations;
  - It two inputs to an S-box differ in precisely one bit, then the output must differ in a minimum of two bits;
  - If two inputs to an S-box differ in their first two bits, but have identical last two bits, the two outputs have to be distinct.

- There have been many other very technical requirements.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. Use two keys, for a double encryption.
2. Use three keys, k_1, k_2 and k_3 to compute

$$c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_3}(w)))$$

How to increase security when encrypting long plaintexts?

$$w = m_1 m_2 \ldots m_n$$

where each m_i has 64-bits.

Choose a 56-bit key k and a 64-bit block c_0 and compute

$$c_i = DES(m_i \oplus c_{i-1})$$

for i = 1,. . . ,n.

The DES CONTROVERSY

1. There have been suspicions that the design of DES might contain hidden "trapdoors'' what allows NSA to decrypt messages.

2. The main criticism has been that the size of the key-space, 2^{56}, is too small for DES to be really secure.

3. In 1977 Diffie+Hellamn suggested that for $ 20 millions one could build a VLSI chip that could search the entire key space within 1 day.

4. In 1993 M. Wiener suggested a machine of the cost $ 100.000 that could find the key in 1.5 days.

WEAKNESSES of DES

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$

- The existence of semi-weak key pairs (k_1, k_2) such that for any plaintext
$$E_{k_1}(E_{k_2}(p)) = p.$$

- The existence of complementation property
$$E_{c(k)}(c(p)) = c(E_k(p)),$$
where c(x) is binary complement of binary string x.

bf MAIN DES MODES of OPERATION

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC (Cipher Block Chaining) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = e_k(c_{i-1} \oplus x_i).$$

OFB (Output Feedback) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a z_0 is chosen, $z_i = e_k(z_{i-1})$ are computed and each x_i is encrypted to get cryptotext $c_i = x_i \oplus z_i$.

CFB (Cipher Feedback) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = x_i \oplus z_i, \text{ where } z_i = e_k(c_{i-1}).$$

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryption;
- OFB mode is used for stream-encryptions that require message authentication;

CTR MODE

Counter Mode – some consider it as the best one.

Key design: $k_i = E_k(n, i)$ for a nonce n;

Encryption: $c_i = x_i \oplus k_i$

This mode is very fast because a key stream can be parallelised to any degree. Because of that this mode is used in network security applications.

8-bit VERSION of the CFB MODE

In this mode each 8-bit piece of the plaintext is encrypted without having to wait for an entire block to be available.

The plaintext is broken into 8-bit pieces: $P = [P_1, P_2, \dots]$.

Encryption: An initial 64-bit block X_1 is chosen and then, for j=1,2,..., the following computation is done:

$$C_j = P_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

where $L_8(X)$ denotes the 8 leftmost bits of X. $R_{56}(X)$ denotes the rightmost 56 bits of X. $X \| Y$ denotes concatenation of strings X and Y.

Decryption:

$$P_j = C_j \oplus L_8(e_k(X_j))$$
$$X_{j+1} = R_{56}(X_j) \| C_j,$$

KILLERS and DEATH of DES

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.
- In 1999 they did that in 24 hours.
- It started to be clear that a new cryptosystem with larger keys is badly needed.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

A Feistel cryptosystem is an iterated cryptosystem mapping 2t-bit plaintext (L_0, R_0) of t-bit blocks L_0 and R_0 to a 2t-bit cryptotext (L_r, R_r), through an r-round process, where $r > 0$.

For $0 < I < r + 1$, the round i maps (L_{i-1}, R_{i-1}) to (L_i, R_i) using a subkey K_i as follows

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

where each subkey K_i is derived from the main key K.

BLOWFISH CRYPTOSYSTEM

- Blowfish is a Feistel type cryptosystem developed in 1994 by Bruce Schneier.
- **Blowfish is more secure and faster than DES.**
- It encrypts 8-bytes blocks into 8-bytes blocks.
- **Key length is a variable 32k, for k = 1, 2, ..., 16.**
- For decryption, Blowfish does not reverse the order of encryption, but it follows it.
- **S-boxes are of key dependence and they, as well as subkeys, are created by repeated execution of Blowfish enciphering transformation.**
- Blowfish has very strong avalanche effect.
- **A follower of Blowfish, Twofish, was one of 5 main candidates for AES.**
- Blowfish can be downloaded free from the B. Schneier web site.

MAIN MODERN ENCRYPTION-DECRYPTION SCHEMES

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.

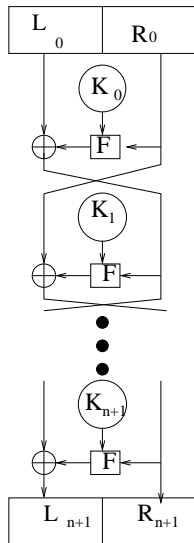- For rounds $i \in \{0, 1, \ldots, n\}$ compute

$$L_{i+1} = R_i; \ R_{i+1} = L_i \oplus F(R_i, k_i)$$

776,13 36then the ciphertext is (R_{n+1}, L_{n+1})

Decryption of (R_{n+1}, L_{n+1}) is done by computing, for $i = n, n-1, \ldots, 0$

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

and (L_0, R_0) is the plaintext

SUBSTITUTION-PERMUTATION ENCRYPTION/DECRYPTION SCHEME

This scheme, known also as **substitution-permutation network**, is an encryption/decryption method/network that performs a series of substitution-permutation layesr of operations composed of S-boxes (substitution boxes) and P-boxes (permutation boxes) as shown in the picture - K_i are keys.
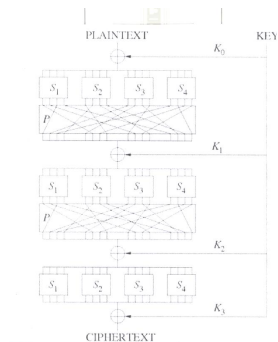
Encryption/decryption system AES discussed next is the most known example of such a system.

V. AES CRYPTOSYSTEM

AES CRYPTOSYSTEM - PREHISTORY

- NIST in USA decided in 1997 to restart encryption standardization process called AES.

- Anyone could make proposal and 15 candidates were accepted in 1998.

- Based on public comments 5 candidates got into second round.

- High security as well as fast speed and low memory requirements on a variety of computing systems were main criteria.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is dedicated to 8-bit microprocessors to encrypt 128-bit messages using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

Motivations and advantages of AES:

- Short code and fast and low memory implementations.
- Simplicity and transparency of the design.
- Variable key length.
- Resistance against all known attacks.

AES - BASIC IDEA

- AES is based on the principle known as substitution-permutation network.
- A 128-bit message is written as a 4×4 matrix of bytes.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:
  1. **SubBytes** - byte-wise substitution defined by a special table of 256 bytes.
  2. **ShiftRows** - a circular shift of i-th row of the matrix by i positions to the left.
  3. **MixColumns** - a linear transformation on each column defined by a 4×4 matrix of bytes.
  4. **AddRoundKey** - bit-wise XOR with a round key defined by another matrix.

PERFORMANCE

AES performs well on a variety of hardware - from 8-bit smart cards to supercomputers.

On a Pentium Pro thhroughput is about 11 MB/s for a 200 MHz processor. On a 1.7 GHz Pentium M, throughput is about 60 MB/s.

On Intel Core i3/i5/i7 CPUs supporting AES-NI instruction set extemsions, throughput can be over 700 MB/s.

The basic data structure of AES is a byte

$$a = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$$

where a_i's are bits, which can be conveniently represented by the polynomial

$$a(x) = a_7 x^7 + a_6 x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0.$$

Bytes can be conveniently seen as elements of the field

$$F = GF(2^8)/m(x), \text{ where } m(x) = x^8 + x^4 + x^3 + x + 1.$$

In the field F, the addition is the bit-wise-XOR and multiplication can be elegantly expressed using polynomial multiplication modulo m(x).

$$c = a \oplus b; \; c = a \bullet b \text{ where } c(x) = [a(x) \bullet b(x)] \mod m(x)$$

MULTIPLICATION in GF(2^8)

Multiplication

$$c = a \bullet b \text{ where } c(x) = [a(x) \bullet b(x)] \mod m(x)$$

in GF(2^8) can be easily performed using a new operation

$$b = \text{xtime}(a)$$

that corresponds to the polynomial multiplication

$$b(x) = [a(x) \bullet x] \mod m(x),$$

as follows

set $c = 00000000$ and $p = a$;

    for $i = 0$ to 7 do
    $c \leftarrow c \oplus (b_i \bullet p)$
    $p \leftarrow \text{xtime}(p)$

Hardware implementation of the multiplication requires therefore one circuit for operation xtime and two 8-bit registers.

Operation $b = \text{xtime}(a)$ can be implemented by one step (shift) of the following shift register:

EXAMPLES

■

<div align="center">

'53' + '87' = 'D4'

</div>

because, in binary,

$$'01010011' \oplus '10000111' = '11010100'$$

what means

$$(x^6 + x^4 + x + 1) + (x^7 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^2$$

■

<div align="center">

'57'• '83' = 'C1'

</div>

Indeed,

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

and

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1)$$
$$\text{mod } (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

■

'57' • '13' = ('57' • '01') ⊕ ('57' • '02') ⊕ ('57' • '10') = '57' ⊕ 'AE' ⊕ '07' = 'FE'

because

$$'57' \bullet '02' = \text{xtime}(57) = 'AE'$$
$$'57' \bullet '04' = \text{xtime}(AE) = '47'$$
$$'57' \bullet '08' = \text{xtime}(47) = '8E'$$
$$'57' \bullet '16' = \text{xtime}(8E) = '07'$$

POLYNOMIALS over GF(2^8)

Algorithms of AES work with 4-byte vectors that can be represented by polynomials of the degree at most 4 with coefficients in GF(2^8).

Addition of such polynomials is done using component-wise and bit-wise XOR. Multiplication is done modulo $M(x) = x^4 + 1$. (It holds x^J mod $(x^4 + 1) = x^{J \bmod 4}$.)

Multiplication of vectors

$$(a_3 x^3 + a_2 x^2 + a_1 x + a_0) \otimes (b_3 x^3 + b_2 x^2 + b_1 x + b_0)$$

can be done using matrix multiplication

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_2 & a_3 & a_0 \\ a_2 & a_3 & a_0 & a_1 \\ a_3 & a_0 & a_1 & a_2 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix},$$

where additions and multiplications ($\cdot$) are done in GF(2^8) as described before. Multiplication of a polynomial a(x) by x results in a cyclic shift of the coefficients.

BYTE-WISE SUBSTITUTION

Byte-wise substitution $b = \text{SubByte}(a)$ is defined by the following matrix operations

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} (a^{-1})_7 \\ (a^{-1})_6 \\ (a^{-1})_5 \\ (a^{-1})_4 \\ (a^{-1})_3 \\ (a^{-1})_2 \\ (a^{-1})_1 \\ (a^{-1})_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

This operation is computationally heavy and it is assumed that it will be implemented by a pre-computed substitution table.

ENCRYPTION in AES

Encryption and decryption are done using state matrices

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

elements of which are bytes.

A byte-matrix with 4 rows and $k = 4, 6$ or 8 columns is also used to write down a key with $D_k = 128, 192$ or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey
4. Final round
   a SubByte
   b ShiftRow
   c AddRoundKey

The final round does not contain MixColumn procedure. The reason being is to be able to use the same hardware for encryption and decryption.

KEY EXPANSION

The basic key is written into the state matrix with 4, 6 or 8 columns.
The goal of the key expansion procedure is to extend the number of keys in such a way that each time a key is used actually a new key is used.

The key extension algorithm generates new columns W_i of the state matrix from the columns W_{i-1} and W_{i-k} using the following rule

$$W_i = W_{i-k} \oplus V,$$

where

$$V = \begin{cases} F(W_{i-1}), & \text{if i mod k} = 0 \\ G(W_{i-1}), & \text{if i mod k} = 4 \text{ and } D_k = 256 \text{ bits,} \\ W_{i-1} & \text{otherwise} \end{cases}$$

and where the function G performs only the byte-substitution of the corresponding bytes. Function F is defined in a quite a complicated way.

STEPS of ENCRYPTION

AddRoundKey procedure adds byte-wise and bit-wise current key to the current contents of the state matrix.

ShiftRow procedure cyclically shifts i-th row of the state matrix by i shifts.

MixColumns procedure multiplies columns of the state matrix by the matrix

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey
3. **do** k+5 - times:
   a) InvSubByte
   b) InvShiftRow
   c) InvMixColumn
   d) AddInvRoundKey
4. Final round
   a) InvSubByte
   b) InvShiftRow
   c) AddInvRoundKey

The goal of the authors was that Rijndael (AES) is K-secure and hermetic in the following sense:

Definition A cryptosystem is K-secure if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible cryptosystems with the same security.

Definition A block cryptosystem is hermetic if it does not have weaknesses that are not present for the majority of cryptosystems with the same block and key length.

Pronunciation of the name **Rijndael** is as "Reign Dahl'' or "rain Doll" or "Rhine Dahl".

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

Efficiency: PKC is much slower than SKC (10 times when software implementations of RSA and DES are compared).

Key sizes: Keys for PKC (2048 bits for RSA) are significantly larger than for SCK (128 bits for AES).

Non-repudiation: With PKC we can ensure, using digital signatures, non-repudiation, but not with SKC.

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes** to send a message **m** using a secret key **k** chosen by the sender, using the public encryption exponent **e** of the receiver, and using the secret decryption exponent **d** of the receiver, as follows:

1. Key **k** is encrypted using **e** and sent as $E_e(k)$
2. Secret description exponent **d** is used to get $k = D_d(E_e(k))$
3. SKC with **k** is then used to encrypt a message
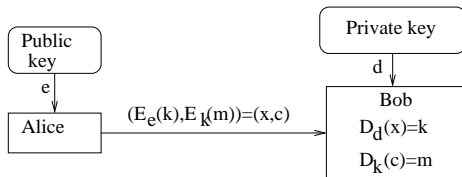
A variety of brute force and also sophisticated methods of attacks have been developed to deal with (also "complex" and without simple mathematical structures) encryption systems.

In case of brute force attacks, an issue is their performance in case they are applied to any encryption algorithm considered as a black box.

Security is measured in such cases in terms of such encryption parameters as the length of the key and the size of message space.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as n oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

This method consists of trying all possible keys exhaustively until the correct key is found.

Exhaustive search can be made more efficient if a probability distribution on keys can be guessed or keys are known to satisfy some relations.

Dictionary attack

Creation of dictionary: For a fixed x and many k values $C_k(x)$ are computed and pairs $(C_k(x), k)$ are inserted into dictionary that is ordered according first item of each pair.

Search If we obtain a $C_k(x)$ value (by chosen plaintext attack), dictionary gives us a list of potential keys.

A generalization for searching for several keys having several values $C_k(x)$ is easy.

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

Differential cryptanalysis: It is assumed that adversary can use the encryption devise as a black box, submitting chosen plaintexts and getting corresponding cryptotext.
The aim of the attack is to get the key.

The basic idea: pairs of random plaintexts are submitted the difference of which is a fixed value a until differences of corresponding cryptotext are at most a fixed value b.

Linear analysis: This is a dual method to differential cryptanalysis invented after discovering anomalies in S-boxes in DES. The idea is not to keep track of difference propagation by the chosen plaintext attack, but to keep track of Boolean information which is linearly obtained by a known plaintext attack: if one gets $(x, c(x))$ pair a statistical analysis of the special Boolean information $L(x, c(x))$ is made and some information on the key is deduced.

ANALYSIS of ATTACKS on AES

So far there have appeared several attacks on AES that are faster than brute force, but only by a minor factor and none of them is feasible.

For AES-128 (AES-192) [AES-256] the key can be recovered with a computational complexity $2^{126.1}$ ($2^{189.7}$) [$2^{254.4}$].

APPENDIX

PUBLIC KEY INFRASTRUCTURE

KEY MANAGEMENT

Secure methods of key management are extremely important. In practice, most of the attacks on public-key cryptosystems are likely to be at the key management levels.

Problems: How to obtain securely an appropriate key pair? How to get other people's public keys? How to get confidence in the legitimacy of others public keys? How to store keys? How to set, extend,... expiration dates of the keys?

Who needs a key? Anyone wishing to sign a message, to verify signatures, to encrypt messages and to decrypt messages.

How does one get a key pair? Each user should generate his/her own key pair. Once generated, a user must register his/her public-key with some central administration, called a certifying authority. This authority returns a certificate.

What are certificates? They are digital documents attesting to the binding of a public-key to an individual or institutions. They allow verification of the claim that a given public-key does belong to a given individual. Certificates help to prevent someone from using a phony key to impersonate someone else. In their simplest form, certificates contain a public-key and a name. In addition they contain: expiration date, name of the certificate issuing authority, serial number of the certificate and the digital signature of the certificate issuer.

CERTIFICATES

Certificates are digital documents attesting to the binding of a public-key to an individual or an institutions.

Certificates allow verification of the claim that a given public-key does belong to a given individual or institution.

Certificates help to prevent someone from using a phony key to impersonate someone else.

In their simplest form, certificates contain a public-key and a name.
In general certificates contain:

- **public-key,**
- **name,**
- **expiration date,**
- **name of the certificate issuing authority,**
- **serial number of the certificate and the digital signature of the certificate issuer.**

HOW are CERTIFICATES used – CERTIFICATION AUTHORITIES

The most secure authentication involves enclosing one or more certificates with every signed message. The receiver of the message verifies the certificate using the certifying authorities public-keys and, being confident of the public-keys of the sender, verifies the message's signature. There may be more certificates enclosed with a message, forming a hierarchical chain, wherein one certificate testifies to the authenticity of the previous certificate. At the top end of a certificate hierarchy is a top-level certifying-authority to be trusted without a certificate.

Example According to the standards, every signature points to a certificate that validates the public-key of the signer. Specifically, each signature contains the name of the issuer of the certificate and the serial number of the certificate.

How do certifying authorities store their private keys?

It is extremely important that private-keys of certifying authorities are stored securely. One method to store the key in a tamper-proof box called a Certificate Signing Unit, CSU.

The CSU should, preferably, destroy its contents if ever opened. Not even employees of the certifying authority should have access to the private-key itself, but only the ability to use private-key in the certificates issuing process.

CSU are for sells

Note: PKCS – Public Key Certification Standards.

- PKI (Public Key Infrastructure) is an infrastructure that allows to handle public-key problems for the community that uses public-key cryptography.
- Structure of PKI

  Security policy that specifies rules under which PKI can be handled.

  Products that generate, store, distribute and manipulate keys.

  Procedures that define methods
  - to generate and manipulate keys
  - to generate and manipulate certificates
  - to distribute keys and certificates
  - to use certificates.

- Authorities that take care that the general security policy is fully performed.

PKI USERS and SYSTEMS

- Certificate holders
- Certificate users
- Certification authorities (CA)
- Registration authorities (RA)
- Revocation authorities
- Repository (to publish a list of certificates, of relocated certificates,...)
- Policy management authorities (to create certification policy)
- Policy approving authorities

SECURITY of CERTIFICATION and REGISTRATION AUTHORITIES

PKI system is so secure how secure are systems for certificate authorities (CA) and registration authorities (RA).

Basic principles to follow to ensure necessary security of CA and RA.

- Private key of CA has to be stored in a way that is secure against intentional professional attacks.
- Steps have to be made for renovation of the private key in the case of a collapse of the system.
- Access to CA/RA tools has to be maximally controlled.
- Each requirement for certification has to be authorized by several independent operators.
- All key transactions of CA/RA have to be logged to be available for a possible audit.
- All CA/RA systems and their documentation have to satisfy maximal requirements for their reliability.

PUBLIC-KEY INFRASTRUCTURE PROBLEMS

Public-key cryptography has low infrastructure overhead, it is more secure, more truthful and with better geographical reach. However, this is due to the fact that public-key users bear a substantial administrative burden and security advantages of the public key cryptography rely excessively on the end-users' security discipline.

Problem 1: With public-key cryptography users must constantly be careful to validate rigorously every public-key they use and must take care for secrecy of their private secret keys.

Problem 2: End-users are rarely willing or able to manage keys sufficiently carefully.

User's behavior is the weak link in any security system, and public-key security is unable to reinforce this weakness.

Problem 3: Only sophisticated users, like system administrators, can realistically be expected to meet fully the demands of public-key cryptography.

MAIN COMPONENTS of PUBLIC-KEY INFRASTRUCTURE

- The **Certification Authority** (**CA**) signs user's public-keys.
  (There has to be a hierarchy of CA, with a root CA on the top.)
- The **Directory** is a public-access database of valid certificates.
- The **Certificate Revocation List** (**CRL**) – a public-access database of invalid certificates. (There has to be a hierarchy of CRL).

Stages at which key management issues arise

- Key creation: user creates a new key pair, proves his identify to CA. CA signs a certificate. User encrypts his private key.
- Single sign-on: decryption of the private key, participation in public-key protocols.
- Key revocation: CRL should be checked every time a certificate is used. If a user's secret key is compromised, CRL administration has to be notified.

MAIN PROBLEMS

- **Authentication of users:** How does a CA authenticate a distant user, when issuing the initial certificate?
  (Ideally CA and the user should meet. Consequently, properly authenticated certificates will have to be expensive, due to the label cost in a face-to-face identity check.)

- **Authenticationg of CA:** Public key cryptography cannot secure the distribution and the validation of the Root CA's public key.

- **Certification of revocation lists:** Timely and secure revocation presents big scaling and performance problems. As a result public-key deployment is usually proceeding without a revocation infrastructure.
  (Revocation is the classical Achilles' Heel of public-key cryptography.)

- **Private key management:** The user must keep his long-lived secret key in memory during his login-session: There is no way to force a public-key user to choose a good password.
  (Lacking effective password-quality controls, most public-key systems are vulnerable to the off-line guessing attacks.)

LIFE CYCLE of CERTIFICATES

Issuing of certificates

- registration of applicants for certificates;
- generation of pairs of keys;
- creation of certificates;
- delivering of certificates;
- dissemination of certificates;
- backuping of keys;

Using of certificates

- receiving a certificate;
- validation of the certificate;
- key backup and recovery;
- automatic key/certificate updating

Revocation of certificates

- expiration of certificates validity period;
- revocation of certificates;
- archivation of keys and certificates.

Pretty Good Privacy

In June 1991 Phil Zimmermann, made publicly available software that made use of RSA cryptosystem very friendly and easy. Bby that he made strong cryptography widely available.

Starting February 1993 Zimmermann was for three years a subject of FBI and Grand Jury investigations, being accused of illegal exporting arms (strong cryptography tools).

William Cowell, Deputy Director of NSA said: "If all personal computers in the world - approximately 200 millions – were to be put to work on a single PGP encrypted message, it would take an average an estimated 12 million times the age of universe to break a single message".

Heated discussion whether strong cryptography should be allowed keep going on. September 11 attack brought another dimension into the problem.

Concerning security we are winning battles, but we are loosing wars concerning privacy.

Four areas concerning security and privacy:

- Security of communications – cryptography
- Computer security (operating systems, viruses, . . .)
- Physical security
- Identification and biometrics

With Google we lost privacy.

HOW IMPORTANT CRYPTOGRAPHIC SYSTEMS get BROKEN

Techniques that are indeed used to break cryptosystems:

By NSA:

- By exhaustive search (up to 2^{80} options).
- By exploiting specific mathematical and statistical weaknesses to speed up the exhaustive search.
- By selling compromised crypto-devices.
- By analysing crypto-operators methods and customs.

By FBI:

- Using keystroke analysis.
- Using the fact that in practice long keys are almost always designed from short guessable passwords.

RSA in PRACTICE

- 660-bits integers were already (factorized) broken in practice.
- 1024-bits integers are currently used as moduli.
- 512-bit integers can be factorized with a device costing 5 K $ in about 10 minutes.
- 1024-bit integers could be factorized in 6 weeks by a device costing 10 millions of dollars.

PATENTABILITY of CRYPTOGRAPHY

- Cryptographic systems are patentable
- Many secret-key cryptosystems have been patented
- The basic idea of public-key cryptography are contained in U.S. Patents 4 200 770 (M. Hellman, W. Diffie, R. Merkle) – 29. 4. 1980 U.S. Patent 4 218 582 (M. Hellman, R. Merkle)

The exclusive licensing rights to both patents are held by "Public Key Partners" (PKP) which also holds rights to the RSA patent.

All legal challenges to public-key patents have been so far settled before judgment.

Some patent applications for cryptosystems have been blocked by intervention of US: intelligence or defense agencies.

All cryptographic products in USA needed export licences from the State department, acting under authority of the International Traffic in Arms Regulation, which defines cryptographic devices, including software, as munition.

Export of cryptography for authentication has not been restricted, Problems were only which cryptography for privacy.

- The main proof of security, that is used in practical cryptography, is the *proof by resilience* - **if a system resists attacks for a number of years, then it may be considered as secure.**

- The common mistake, that is repeated through the history of cryptography, is that a "complicated cryptosystem" must be secure.