Lecture 6

Block and stream ciphers

Nicola Laurenti October 16, 2020

Lecture 6— Contents

Block ciphers

A block cipher is an encryption system in which both the plaintext and ciphertext have fixed length

$$\mathcal{M} = \mathcal{A}_u^{\ell_u} \quad , \quad \mathcal{X} = \mathcal{A}_x^{\ell_x}$$

Often, they also share the same alphabet and same length, and in this case

stesso alfabeto $\mathcal{M} = \mathcal{X} = \mathcal{A}^{\ell}$ stessa lunghezza

Typically, the key is also represented as a fixed length vector with the same alphabet, that is

$$\mathcal{K} = \mathcal{A}^{\ell_k}$$

Simple examples

Example (Transposition ciphers)

$$\mathcal{M} = \mathcal{X} = \mathcal{A}^\ell$$

$$E_k : x = [x_1, \ldots, x_\ell]$$

$$u = [u_1, \ldots, u_\ell]$$

$$x_i = u_{e_k(i)} \quad , \quad i = 1, \ldots, \ell$$

$$e_k : \{1, \ldots, \ell\} \mapsto \{1, \ldots, \ell\}$$

invertible, i.e. a permutation

The class $\{e_k\}$ of all permutations of ℓ elements has cardinality $K = \ell!$
We can take $\mathcal{K} = \{1, \ldots, K\}$.

LA RIVISTA CHE VANTA INNUMEREVOLI TENTATIVI D'IMITAZIONE!

LA SETTIMANA ENIGMISTICA

ESCE IL GIOVEDÌ

10 Luglio 2014
N. 4294 Anno III
Euro 1,50 (in Italia)

Periodico di parole crociate, rebus, enigmi, paramenpi, varietà, umorismo, ecc.

6288. IL TIPOGRAFO DISTRATTO

Un tipografo, nel comporre questo aneddoto, ha commesso numerosi refusi. Lasciamo ai Lettori il compito di correggerlo, anagrammando le parole prive di significato.

Il berlece mocipatoldi screnafe Talleyrand era stato ottivani a neac da una daniblonon che, per zoangrina o per rodisnizeta, non gli aveva tensagosa a lovata il topso che gli taptasve. Uno dei pentersi fece toenar l'errore alla radonpa di asca, e lei si trafôfte a steperrane le proprie susce all'illustre petiso.

– Ma mia cara rosniga – brétati lui mnetardefde, – quonvedu io mi esadi, eloluq è il spoto d'onore!

Vedete poi la soluzione pubblicata a pagina 46

Simple examples

Example (Linear ciphers)

Starting from a finite field $(\mathbb{F}, +, \cdot)$

$$\text{choose } \mathcal{A}_u = \mathcal{A}_x = \mathbb{F}$$
$$\text{then } \mathcal{M} = \mathbb{F}^{\ell_u} \quad , \quad \mathcal{X} = \mathbb{F}^{\ell_x} \quad \text{are linear spaces on } \mathbb{F}$$

Encryption and decryption are linear maps between $\mathbb{F}^{\ell_u} \leftrightarrow \mathbb{F}^{\ell_x}$

$E_k : x = M_k u \quad , \quad M_k \in \mathbb{F}^{\ell_x \times \ell_u}$ a matrix injective map $\Leftrightarrow$ full column rank ℓ_u

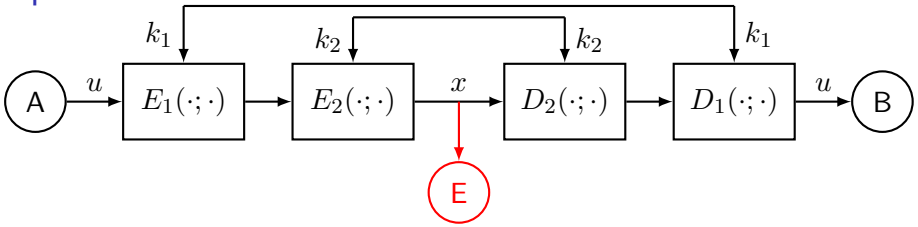
$D_k : u = M_k^{-1} x \quad , \quad M_k^{-1} \in \mathbb{F}^{\ell_u \times \ell_x}$ M_k^{-1} is a left inverse of M_k, i.e. $M_k^{-1} M_k = I_{\ell_u}$

The class $\{M_k\}$ of all $\ell_x \times \ell_u$ matrices with entries from $\mathbb{F}$ and rank ℓ_u has cardinality $K \lessapprox |\mathbb{F}|^{\ell_x \ell_u}$.
We can take $\mathcal{K} = \mathbb{F}^{\ell_x \times \ell_u}$ and $k = M_k$ (written as a vector, e.g., columnwise).

Building a robust cipher

Substitution, transposition and linear ciphers are not robust

Cipher concatenation



Let $\mathcal{S}_1 = (\mathcal{M}_1, \mathcal{X}_1, \mathcal{K}_1, E_1, D_1, p_{k_1}, p_{u_1})$ and $\mathcal{S}_2 = (\mathcal{M}_2, \mathcal{X}_2, \mathcal{K}_2, E_2, D_2, p_{k_2}, p_{u_2})$ be two ciphers such that $\mathcal{X}_1 \subseteq \mathcal{M}_2$. Their concatenation is the cipher $\mathcal{S} = (\mathcal{M}, \mathcal{X}, \mathcal{K}, E, D, p_k, p_u)$ where

$$\mathcal{M} = \mathcal{M}_1 \quad , \quad \mathcal{X} = \mathcal{X}_2 \quad , \quad \mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$$
$$E(k,u) = E([k_1,k_2],u) = E_2(k_2, E_1(k_1,u))$$
$$D(k,x) = D([k_1,k_2],x) = D_1(k_1, D_2(k_2,x))$$
$$p_k(c) = p_{k_1k_2}(c_1,c_2) \stackrel{?}{=} p_{k_1}(c_1)p_{k_2}(c_2) \quad , \quad p_u(a) = p_{u_1}(a)$$

"Good" and "bad" concatenations

Concatenating many substitution ciphers does not increase security, as substitution encryption maps are a group under composition

$$\forall k_1, k_2, \exists k_3 \ : \ E_{k_2} E_{k_1} = E_{k_3}$$

The same holds for concatenating all transpositions, or all linear maps

However, if substitution, transposition and linear are alternated repeatedly, the resulting cipher is robust

$$E_k = L_{k_n} S_{k_{n-1}} \cdots T_{k_6} L_{k_5} S_{k_4} T_{k_3} L_{k_2} S_{k_1}$$

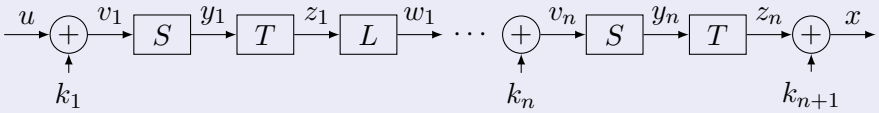
A serious example: Advanced Encryption Standard (AES)

It is a (S, T, L) iterated cipher with $n = 10, 12$ or 14 rounds and

$$\mathcal{M} = \mathcal{X} = \mathcal{K}' = \mathbb{F}^{\ell} \quad , \quad \mathbb{F} = \mathrm{GF}(2^8)(\text{byte field}) \quad , \quad \ell = 16$$

$$\mathcal{K} = \mathbb{F}^{\ell_k} \quad , \quad \ell_k = 16, 24, 32$$

Encryption



Decryption is performed by the inverse blocks in reverse order

A serious example: Advanced Encryption Standard (AES)

Efficient software implementation

subkey generation

$$k_i \in \mathcal{K}' \quad , \quad g : \mathcal{K} \mapsto (\mathcal{K}')^{n+1}$$

substitution

$$v_i = (v_{i,1}, \ldots, v_{i,\ell}) \quad , \quad y_i = (y_{i,1}, \ldots, y_{i,\ell}) \quad , \quad y_{i,j} = f(v_{i,j})$$

based on $v_{i,j}^{-1}$ inverse in $\mathbb{F}$ but stored in a lookup table

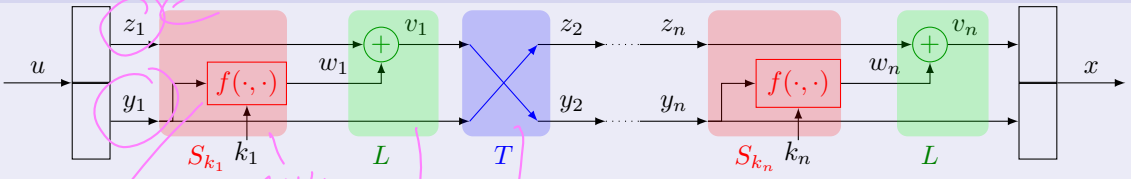
transposition write (by columns) vector y_i to 4×4 matrix Y_i, then ciclically shift j-th row by $j-1$, read matrix by columns into vector z_i (e.g., $z_{i,1} = y_{i,1}$, $z_{i,2} = y_{i,14}$)

linear write (by columns) vector z_i to 4×4 matrix Z_i, convolve cyclically each column with $[2, 1, 1, 3]$

Feistel ciphers

A Feistel cipher is a binary block cipher with $\mathcal{M} = \mathcal{X} = \mathbb{B}^{2\ell}$ that is based on the following n-round (S, T, L) iterated structure $E_k = LS_{k_n} \cdots TLS_{k_2}TLS_{k_1}$

Encryption



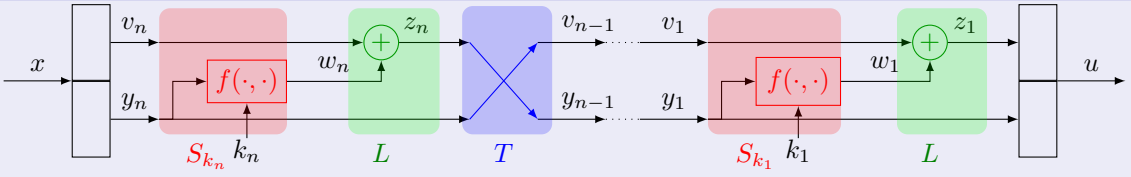
1. First the plaintext u is split into two ℓ-bit blocks y_1 and z_1
2. Then at each round i the following transformation are applied

   substitution $S : \mathcal{K}' \times \mathbb{B}^{2\ell} \mapsto \mathbb{B}^{3\ell}$, $S_{k_i}(y_i, z_i) = [y_i, w_i, z_i]$, $w_i = f(k_i, y_i)$
   linear tf $L : \mathbb{B}^{3\ell} \mapsto \mathbb{B}^{2\ell}$, $L(y_i, w_i, z_i) = [y_i, v_i]$, with $v_i = w_i + z_i$
   transposition $T : \mathbb{B}^{2\ell} \mapsto \mathbb{B}^{2\ell}$, $T(y_i, v_i) = [v_i, y_i] = [y_{i+1}, z_{i+1}]$, $i \neq n$
3. Last, y_n and v_n are concatenated to make the ciphertext x

Feistel ciphers

A Feistel cipher can be decrypted by using the same operations and in the same order, except for the inversion of the key sequence, i.e.: $D_k = LS_{k_1}TLS_{k_2}\cdots TLS_{k_n}$

Decryption



1. Split x into y_n and v_n
2. Then at each round i running backwards (from n to 1)

$$S_{k_i}(y_i, v_i) = [y_i, w_i, v_i] \ , \quad \text{with } w_i = f(k_i, y_i)$$
$$L(y_i, w_i, v_i) = [y_i, z_i] \ , \quad \text{with } z_i = w_i + v_i$$
$$T(y_i, z_i) = [z_i, y_i] = [y_{i-1}, v_{i-1}] \ , \quad i \neq 1$$

3. Last, y_1 and z_1 are concatenated to make the plaintext u

Example: Data Encryption Standard (DES)

- A Feistel cipher with binary keys and lengths $\ell_k = 56$, $\ell_u = \ell_x = 64$, $\ell = 32$, using $n = 16$ rounds
- Designed by IBM in 1977 for the US NSA
- Efficient hardware implementation

Security features

- Moderately secure against brute force (key too short even then)
- Careful design of the round function $f(\cdot, \cdot)$ avoiding linear and differential cryptanalysis (only discovered in the 90's)

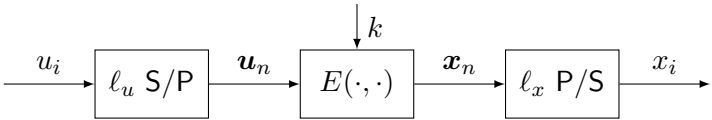
Block encryption of long messages

In order to use block ciphers for longer (ideally infinite) messages, one can either:

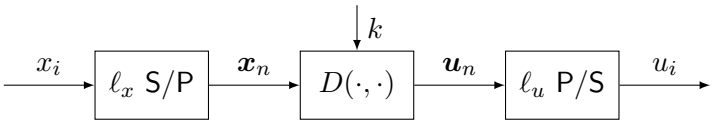
- convert the input plaintext message into a sequence of (non overlapping) blocks, encrypt the blocks, and recombine the sequence of output cipher blocks into the ciphertext (e.g., ECB, CBC)
- use a block cipher to generate a long enough keystream message and combine it with the plaintext to generate the ciphertext (e.g., CFB, OFB, CTR)

Electronic codebook mode (ECB)

Encryption



Decryption



Pros

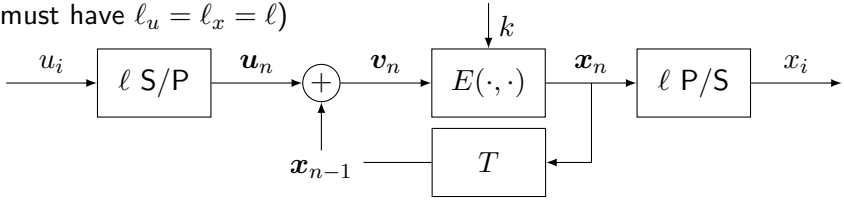
simple to implement
no error / loss propagation

Cons

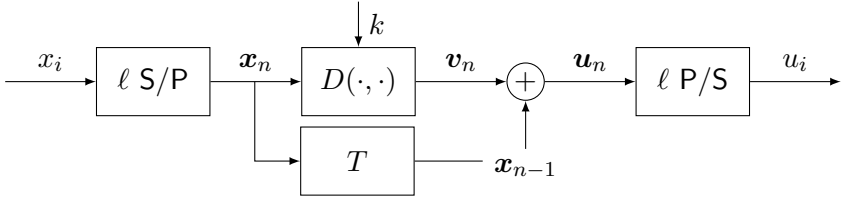
allows frequency analysis attack
targeted modification attacks

Cipher block chaining mode (CBC)

Encryption (must have $\ell_u = \ell_x = \ell$)



Decryption



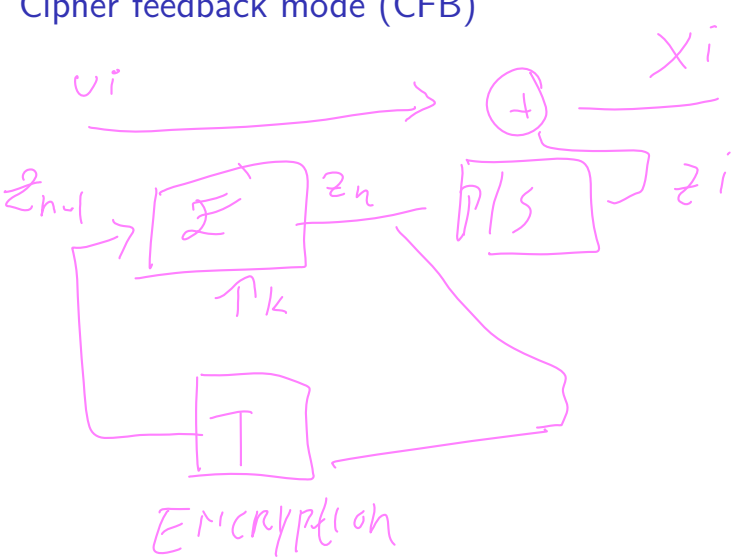
Pros

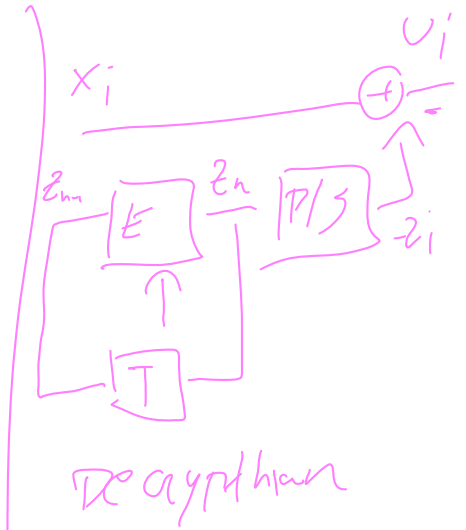
no frequency analysis
limited error propagation

Cons

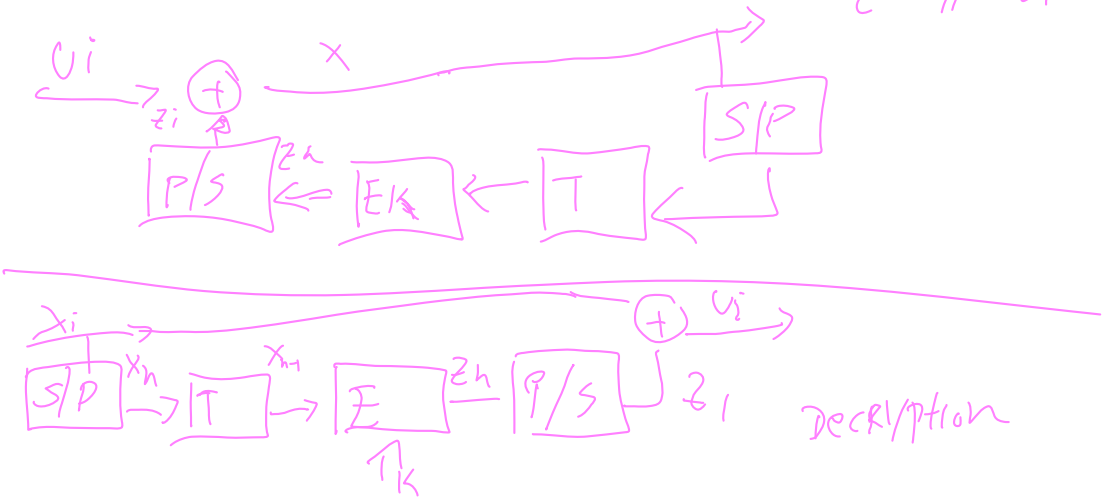
predictable modification attacks

Cipher feedback mode (CFB)

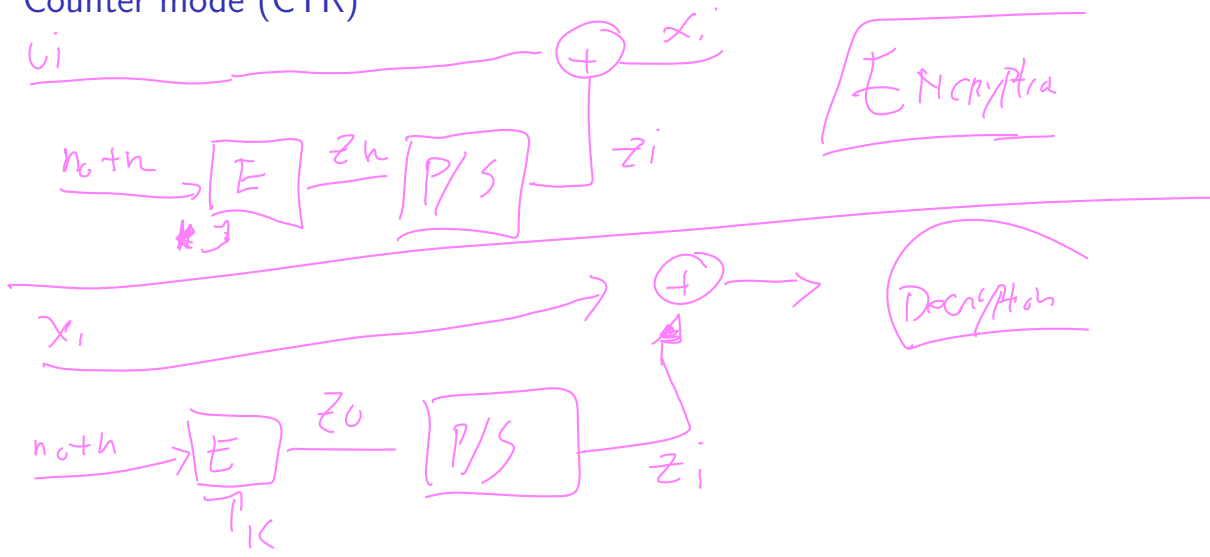




Output feedback mode (OFB)



Counter mode (CTR)



General model for stream ciphers

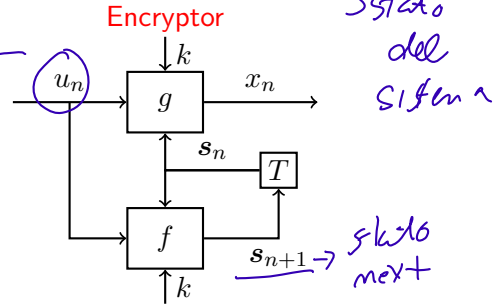
In stream ciphers E_k can be viewed as a (non linear) dynamical system discrete-time.
Block cipher $\rightarrow$ fixed length input.
Stream cipher $\rightarrow$ indefinite possibly infinite length input.

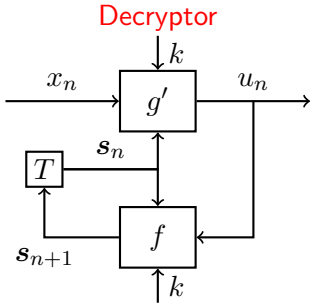
↓
Less
Computation

Give possibly

State space representation of E_k and D_k

: $\boldsymbol{s}_n \in \mathcal{S}$: state of the system at 'time' n, state space.





$$\begin{cases} \boldsymbol{s}_{n+1} = f_k(n, \boldsymbol{s}_n, u_n), & \text{update} \\ x_n = g_k(n, \boldsymbol{s}_n, u_n), & \text{output} \end{cases}$$

$$\begin{cases} \boldsymbol{s}_{n+1} = f_k(n, \boldsymbol{s}_n, u_n), & \text{update} \\ u_n = g'_k(n, \boldsymbol{s}_n, x_n), & \text{output} \end{cases}$$

The receiver must know $k, \boldsymbol{s}_0$ (key, initialization value), both secret. The system ensures invertibility: $D_k = E_k^{-1}$, if $g'_k(n, \boldsymbol{s}_n, \cdot)$ inverts $g_k(n, \boldsymbol{s}_n, \cdot)$.

Additive stream ciphers

Very wide class of stream ciphers,

$$\mathcal{A}_u = \mathcal{A}_x = \mathcal{A}_z = \mathbb{G}$$

with $\mathbb{G}$ a group, where

$$\begin{cases} \boldsymbol{s}_{n+1} = f_k(n, \boldsymbol{s}_n), & \text{update does not depend on plaintext} \\ x_n = u_n + h_k(n, \boldsymbol{s}_n), & \text{output depends additively on plaintext} \end{cases}$$

with

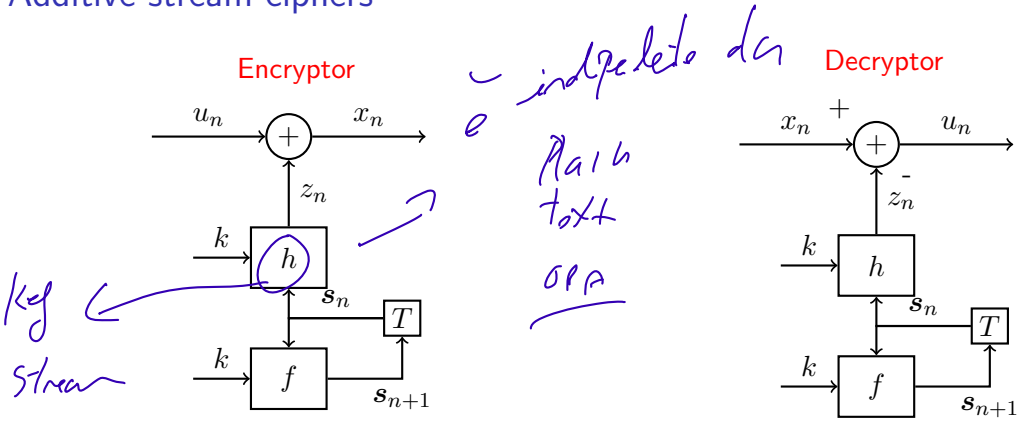
$$z_n = h_k(n, \boldsymbol{s}_n) \quad \text{key stream}$$

The decryption is:

$$\begin{cases} \boldsymbol{s}_{n+1} = f_k(n, \boldsymbol{s}_n) \\ u_n = x_n - h_k(n, \boldsymbol{s}_n) \end{cases}$$

$\boldsymbol{s}_0$ needs to be secret, but often $\boldsymbol{s}_0 = \mu(k)$ (function of k) and then $f_k = f$, $h_k = h$ can be independent of the key.

Additive stream ciphers



Problem: keystream reuse

KPA: two messages u_n, u'_n are encrypted with the same z_n, ie $x_n = u_n + z_n$ and $x'_n = u'_n + z_n$.
If the attackers knows u_n, x_n, x'_n will be able to decrypt $u'_n = x'_n - z_n = x'_n - x_n + u_n$

Example: Rivest Cipher 4 (RC4)

$$\mathcal{A}_u = \mathcal{A}_x = \mathcal{A}_z = \mathcal{A}_k = \mathcal{A}_s = \mathbb{G} \quad , \quad M = |\mathbb{G}| = 2^8$$

$$\mathcal{K} = \mathcal{A}_k^\ell = \mathbb{G}^\ell \quad , \quad \mathcal{S} = \mathbb{G}^{M+1}, \ \boldsymbol{s}_n = [s_n(0) \dots s_n(M)]$$

Key stream generation

$$h \ : \ z_n = s_n(s_n(M) + n + 1)$$

depends on current state and time, not on key. Note that index + is in $\mathbb{G}$

State update

$$f \ : \ \begin{cases} s_n(M) = s_{n-1}(M) + s_{n-1}(n+1) \\ s_n(s_n(M)) = s_{n-1}(n+1) \\ s_n(n+1) = s_{n-1}(s_n(M) \\ s_n(m) = s_{n-1}(m), \quad m \neq M \end{cases}$$

Example: Rivest Cipher 4 (RC4)

State initialization

$$\begin{cases} s_{-M}(m) = m, & m = 0, \dots, M-1 \\ s_{-M}(M) = 0 \end{cases}$$

then, for $n = -M+1, \dots, 0$

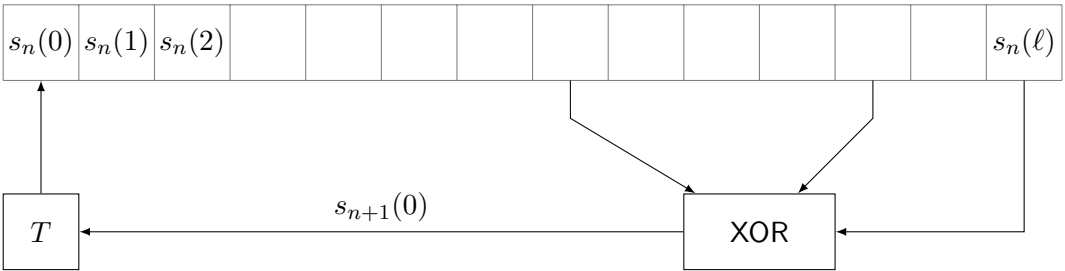
$$f' : \begin{cases} s_n(M) = s_{n-1}(M) + k_{n+M} + s_{n-1}(s_{n-1}(M)) \\ s_n(n) = s_{n-1}(n) + k_{n+M} + s_{n-1}(n)) \end{cases}$$

f' is the only function depending on k, hence $\boldsymbol{s}_0 = \mu(k)$

RC4 in WEP

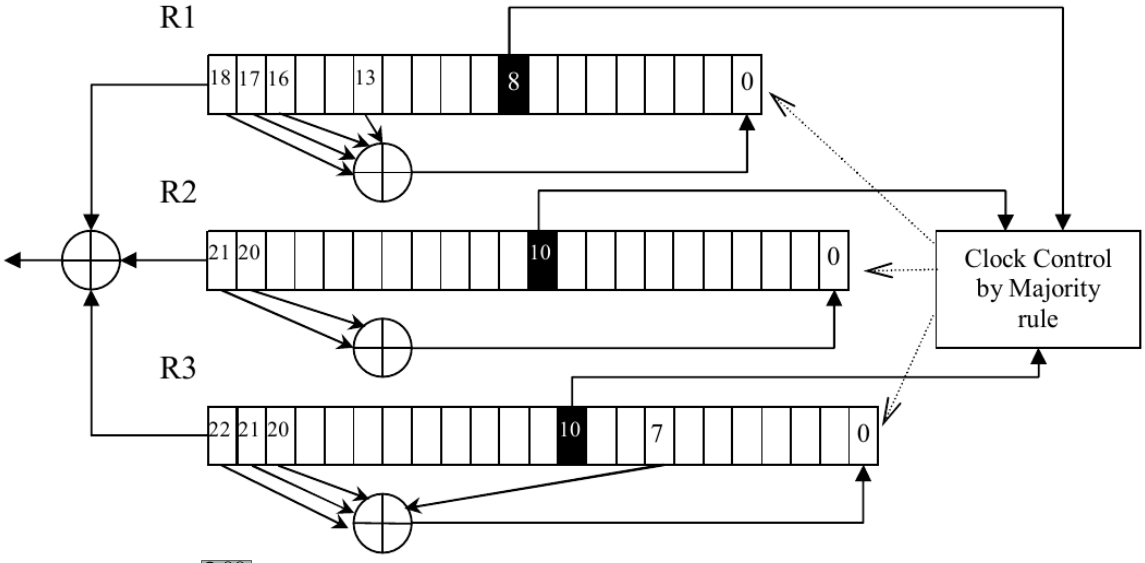
RC4 is used in WEP with $\ell = 5$, but $k = [k', v_0]$ where $\ell_{k'} = 3$ bytes, secret; $\ell_{v_0} = 2$ bytes, public.
Reusing k yields the same $\boldsymbol{s}_0$ and so the same z_n.

Stream ciphers based on Linear Feedback Shift Registers (LFSR)



- the state $\boldsymbol{s}_n$ is the content of one (or more) LFSR
- state update can be computed efficiently
- some nonlinearity should be introduced in the output or update function to avoid easy reconstruction of $\boldsymbol{s}_n$ from te keystream, e.g., by combining more LFSRs

The GSM cipher A5/1



The GSM cipher A5/1

A5/1 is a binary stream cipher

$$\mathcal{A}_u = \mathcal{A}_x = \mathcal{A}_z = \mathcal{A}_k = \mathcal{A}_s = \mathbb{B} = \{0, 1\}$$

The global state comprises the state of the 3 registers (19-bit, 22-bit, and 23-bit), both state and key are 64-bit long

$$\mathcal{K} = \mathcal{A}_k^{\ell_k} = \mathbb{B}^{\ell_k} \quad , \quad \mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{S}_3 \quad , \quad \boldsymbol{s}_n = [\boldsymbol{s}_{1,n}, \boldsymbol{s}_{2,n}, \boldsymbol{s}_{3,n}] \quad , \quad \mathcal{S}_i = \mathbb{B}^{\ell_i}$$

$$\boldsymbol{s}_{i,n} = [s_{i,n}(0), \ldots, s_{i,n}(\ell_i - 1)] \quad , \quad \ell_1 = 19, \ell_2 = 22, \ell_3 = 23 \quad , \quad \ell_s = \ell_1 + \ell_2 + \ell_3 = 64$$

Key stream generation

$$h \, : \, z_n = s_{1,n}(\ell_1 - 1) \oplus s_{2,n}(\ell_2 - 1) \oplus s_{3,n}(\ell_3 - 1)$$

depends on current state (XOR the last bit from each register), not on key.

State update each register i advances only if $s_{i,n}(c_i) = s_{j,n}(c_j)$ for some other $j \neq i$
(at each step either all or only two registers advance)

The GSM cipher A5/1: vulnerabilities

Specific vulnerabilities

- State update of A5/1 is not one-to-one
- Long time with the same BSC, states will concentrate
- 64 bit key / state are too short

→

Biased birthday state guessing attack

1. precompute the 64-bit outputs that correspond to the most likely states
2. observe until any of them appears in the actual transmission
3. the state is known

Summary

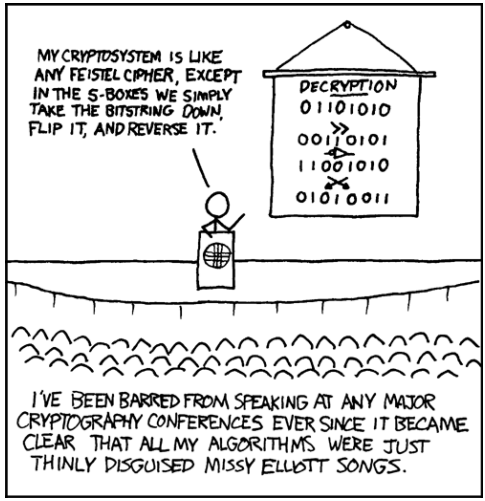
In this lecture we have:

- introduced block ciphers, with AES and DES as examples
- discussed operating modes for block ciphers
- introduced stream ciphers, with RC4 and A5/1 as examples

Assignment

- class notes
- textbook, §4.1–§4.2, §4.5–§4.6 (except pages 111-114)

End of lecture



Cryptography, reproduced from xkcd URL: xkcd.com/153