Stream Ciphers

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it

Last version: 2021-03-02

1

Stream Ciphers

ONE-TIME PAD

2

One Time Pad PERFECT CIPHER

(Vernam cipher, 1917)

UNIVERSITÀ DI PISA

- Definition MESSAGE $0 \dots t-1$ $\oplus \Rightarrow$ $0 \dots t-1$ TESTO CIFRATO
  - Assumptions CHIAVE $0 \dots t-1$
    - Let x be a t-bit message, i.e., $x \in \{0,1\}^t$
    - Let k be a t-bit key stream, $k \in \{0, 1\}^t$, where each bit is truly random chosen
    - The key is only known to the legitimate communicating partners and is used just once
  - Encryption → BITWISE EXCLUSIVE OR
    - $y_i = m_i \oplus k_i$ i.e., $y_i = m_i + k_i \bmod 2$
  - Decryption
    - $x_i = c_i \oplus k_i$, i.e., $x_i = y_i + k_i \bmod 2$
  - Consistency property can be easily proven

$c_i \oplus k_i = (c_i + k_i) \bmod 2 = (m_i + k_i + k_i) \bmod 2 = (m_i + 2k_i) \bmod 2 = m_i$

$c_i = m_i \oplus k_i = (m_i + k_i) \bmod 2 =$

m_i	k_i	c_i
0	0	0
0	1	1
1	0	1
1	1	0

March 21 Stream Ciphers 3

3

Why $\oplus$ is a good encryption function?

UNIVERSITÀ DI PISA

- **Theorem.** DIMOSTRAZIONE SU NOTE
  - Let X be a random variable on $\{0, 1\}^n$, and K an independent uniform variable on $\{0,1\}^n$.
  - Then, $Y = X \oplus K$ is uniform on $\{0,1\}^n$.
  - **Proof.** (for n = 1)

USATA MOLTO SPESSO!

March 21 Stream Ciphers 4

4

x

k

y

$y_0 = (x_0 + k_0) \bmod 2$ → L'AVVERSARIO LE CONOSCE, È EQUAZIONE IN 1 VARIABILE CONOSCIUTA 2 SCONOSCIUTE → NON RISOLVIBILE

$y_1 = (x_1 + k_1) \bmod 2$

$y_t = (x_t + k_t) \bmod 2$

OTP has perfect secrecy

CIÒ VALE SOLO SE TUTTI GLI K SONO RANDOM COSÌ CHE NON CI SONO RELAZIONI TRA I BIT

- **THM.** OTP has perfect secrecy $\boxed{\text{iff}}$

1. The key stream k_i is truly random → MA SE NON VALE HO UN SISTEMA RISOLVIBILE
2. The key stream k_i is only known to the communication parties
3. Every key stream k_i is used just once → NON DEVE ESSERE RIUTILIZZATA

OTP has perfect secrecy: intuition

- **c[i] = m[i] + k[i] mod 26**
- **m = "SUPPORT JAMES BOND"**

m =	S	U	P	P	O	R	T	J	A	M	E	S	B	O	N	D
k =	W	C	L	N	B	T	D	E	F	J	A	Z	G	U	I	R
c =	O	W	A	C	P	K	W	N	F	V	E	R	H	I	V	U

16 CHARS

K

K

TUTTE LE POSSIBILI CHIAVI

26^{16} → TUTTI I POSSIBILI MESSAGGI

→ NON C'È MODO DI CAPIRE QUALE SIA IL VERO MESSAGGIO PERCHÉ LA CHIAVE È COMPLETAMENTE CASUALE

c =	O	W	A	C	P	K	W	N	F	V	E	R	H	I	V	U
k' =	M	W	L	J	V	T	S	E	F	J	A	Z	G	U	I	R
m =	C	A	P	T	U	R	E	J	A	M	E	S	B	O	N	D

Pros and Cons of OTP - Pros

- Unconditionally secure
  - A cryptosystem is unconditionally or information-theoretically secure if it cannot be broken even with infinite computational resources
- Very fast enc/dec
- Only one key maps m into c

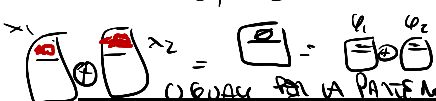
8

2 MESSAGGI x_1 e x_2, USO LA STESSA CHIAVE k

$y_1 = x_1 \oplus k$

$y_2 = x_2 \oplus k$

$\Rightarrow y_1 \oplus y_2 = x_1 \oplus \cancel{k} \oplus x_2 \oplus \cancel{k} = \boxed{x_1 \oplus x_2}$ MALE!



UGUALI PER LA PARTE ROSSA

RACCOGLIENDO SU TESTO CIFRATO OTTENGO INFORMAZIONI SU TESTO NON CIFRATO

Pros and Cons of OTP - Cons

MESSAGGI LUNGHI

↳ CHIAVI LUNGHE

- Long keys: unpractical!
  - Key len == msg len

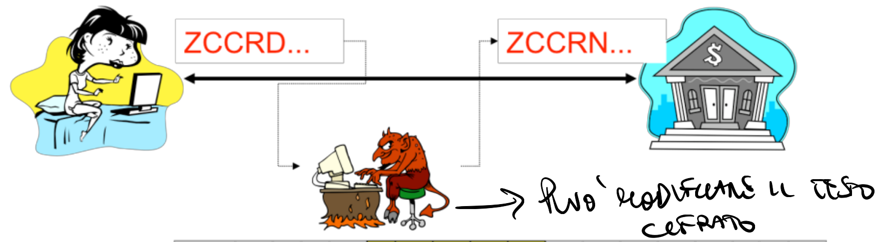
LO RENDE NON SICURO

- Keys must be used once: avoid two-time pad!
  - Let C1 = M1 xor K and C2 = M2 xor K => C1 xor C2 = M1 xor M2 => Redundancies of M1, M2 can be exploited (e.g., English and ASCII)
- A Known-PlainText attack breaks OTP
  - Given (m, c) => k = m xor c
- OTP is malleable
  - Modifications to cipher-text are undetected and have predictable impact on plain-text

9

OTP is malleable

m =	D	A	R	E	C	E	N	T	O	E	U	R	O	A	B	O	B
k =	W	C	L	N	B	T	D	E	F	J	A	Z	G	U	I	R	X
c =	Z	C	C	R	D	X	Q	X	T	N	U	Q	U	U	J	F	Y



c' =	Z	C	C	R	N	B	O	P	J	N	U	Q	U	U	J	F	Y
k =	W	C	L	N	B	T	D	E	F	J	A	Z	G	U	I	R	X
m =	D	A	R	E	M	I	L	L	E	E	U	R	O	A	B	O	B

Malleability

- Malleability
  - A crypto scheme is said to be malleable if the attacker is capable of transforming the ciphertext into another ciphertext which leads to a known transformation of the plaintext
    - The attacker does not decrypt the ciphertext but (s)he is able to manipulate the plaintext in a predictable manner

On OTP malleability

UNIVERSITÀ DI PISA

- Attack against integrity
  - Alice sends Bob: $c = p \oplus k$
  - The adversary
    - intercepts c and
    - transmits Bob $c' = c \oplus r$, with r called perturbation
  - Bob
    - receives c' and
    - Computes $p' = c' \oplus k = c \oplus r \oplus k = p \oplus k \oplus r \oplus k$ so obtaining $p' = p \oplus r$
    - The perturbation goes undetected
    - The perturbation has a predictable impact on the plaintext

Stream Ciphers

STREAM CIPHERS

Making OTP practical (1/3)

UNIVERSITÀ DI PISA

- Idea: replace the random key stream by a pseudo-random key stream
- Pseudo Random Generator G is an efficient and deterministic function

LA VERA CHIAVE E' QUESTA

SEED

KEYSTREAM

$$G : \{0,1\}^s \rightarrow \{0,1\}^n, n >> s$$

|SEED| << |KEYSTREAM|

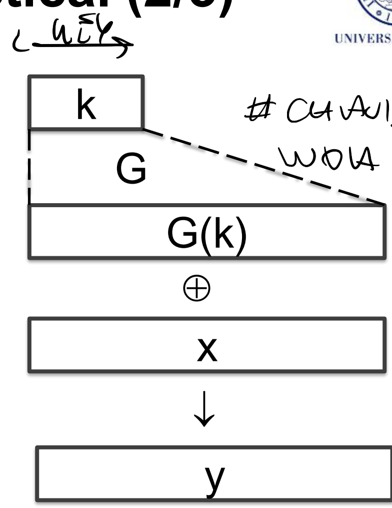
Seed space **Key-stream space**

The key stream is computed from a seed

Making OTP practical (2/3)

UNIVERSITÀ DI PISA

Encryption: y = G(k) ⊕ x
Decryption: x = G(k) ⊕ y



CHIAVE << # MSG VIOLA LA REGOLA DI SHANNON

- Key *k* is a small secret (e.g., 100 bits)
- G is pseudo-random so sndr & rcvr generate the same key stream

Making OTP practical (3/3)

- Is OTP-modified (stream cipher) still perfect?
  - NO! #keys < #msg => Shannon's theorem violated
    - We need a new definition of security!
- Security will depend on the specific PRG
  - PSEUDO
  - PRG must look random, i.e., indistinguishable from a TRG for a limited adversary
    - It must be computationally unfeasible to distinguish PRNG output from a TRG output
  - TRULY RANDOM GENERATOR
  - A new definition of security is necessary: computational security

Computational security

UN MODO DIVERSO PER DIRE CHE L'AVVERSARIO HA UNA POTENZA CALCOLATA

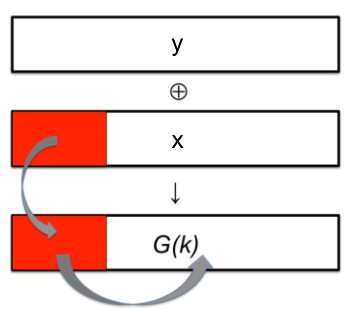
- Definition
  - A cryptosystem is computationally secure if the best known algorithm for breaking it requires at least *t* operations
  - Cons
    - What is the best known attack?
    - The best we can do it to design cryptosystem for which it is *assumed* that they are computationally secure

Computational security

- Cons
  A. What is the best known attack?
  B. Even if a lower bound on the complexity of one attack is known, we don't know whether any other, more powerful attacks, are possible
- The best we can do it to design cryptosystem for which it is *assumed* that they are computationally secure

Why we need predictability

- If PRG is predictable, a stream cipher is not secure! → NON PREDICIBILE
  - Assume an adversary is able to determine a prefix of x then
  - Then, (s)he can compute a prefix of the key stream
  - If G is predictable, (s)he can compute the rest of the key stream and thus decrypt y



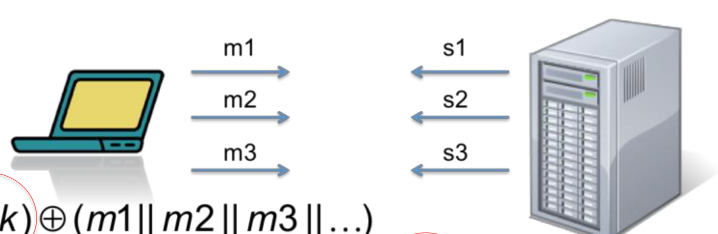
Stream ciphers

STATE OF THE ART AND CASE STUDIES

USA STREAM CIPHER PER CLIENT-SERVER COMMUNICATION

MS-PPTP (Windows NT)

UNIVERSITÀ DI PISA



$$G(k) \oplus \underbrace{(m1 \| m2 \| m3 \| \ldots)}$$

FLOW OF MESSAGES (STREAM)

CONDIVIDONO LA CHIAVE

$$G(k) \oplus \underbrace{(s1 \| s2 \| s3 \| \ldots)}_{\text{STREAM}}$$

Two-time pad!

$K = (K_{c \to s}, K_{s \to c})$ is the right way!

UNA CHIAVE PER CLIENT-SERVER
UNA PER SERVER-CLIENT

2 GRANDI VULNERABILITÀ: IV || k DOVREBBE ESSERE 128 BIT

SIZEOF(IV || k) = 128 BIT
SIZEOF(k) = 104 BIT
SIZEOF(IV) = 24 BIT

$\underbrace{IV \| k}_{\text{SEED}} \to \boxed{PRG} \to z \oplus m, CRC(m)$

$\downarrow$

$\boxed{IV}\ \boxed{CIPHERTEXT}$

AL MASSIMO 2^{24} MESSAGGI DIVERSI, DOPO DICHE
↓
TWO-TIME PAD

IL WRAP-AROUND

PRG = RC4

$IV \| k \to \boxed{PRG\ (RC4)} \to z$

↑
QUESTE CHIAVI SONO CORRELATE!

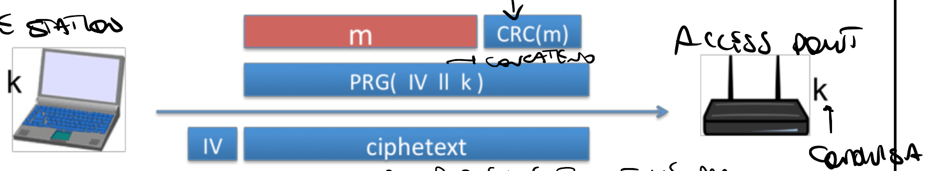
802.11b WEP

UNIVERSITÀ DI PISA

REDUNDANCY CODE

MOBILE STATION

ACCESS POINT



INITIALIZATION VECTOR → PER PREVENIRE TWO-TIME PAD

- A new IV for each new message
  - Key is fixed (104-bits)
  - IV avoids 2TP

PER OGNI m CREIAMO UN NUOVO IV -

- Length of IV: 24 bits (in the standard!)
  - Repeated IV after $2^{24} \approx$ 16M frames
  - On some 802.11 cards IV resets to 0 after power cycle

PUÒ ESSERE UN COUNTER CHE VIENE QUINDI RESETTATO AD OGNI SPEGNIMENTO

March 21 Stream Ciphers 22

22

→ LO STANDARD NON DICE COME GENERARE IV

RC4 NON DEVE AVERE CHIAVI CORRELATE
↓
DIVENTA ATTACCABILE

0 || k
1 || k
⋮

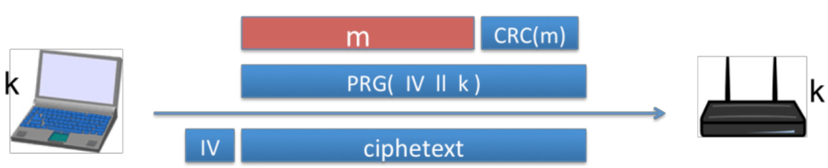
FMS 2001

ATTACCO CHE SI PUÒ FARE DOPO ~10^6 FRAME CRIPTATI

ORA BASTANO 40000 FRAME

802.11b WEP

UNIVERSITÀ DI PISA



Key for frame #1: 1||k
Key for frame #2: 2||k
Key for frame #3: 3||k
…

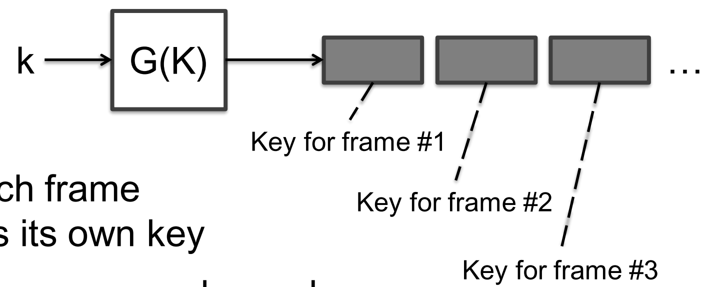
- Related keys, not random
- FMS 2001 attack can recover *K* in 10^6 frames (now 40 Kframes)
- **Avoid related keys!**

March 21 Stream Ciphers 23

23

802.11b: WEP

- A better construction



- Each frame has its own key
- Keys are pseudo-random

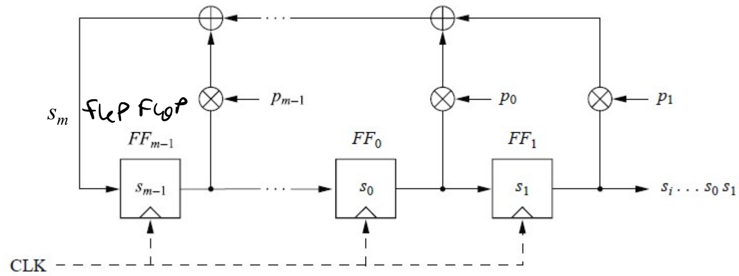
RC4

- RC4 (1987)
  - Used in HTTPS and WEP
  - Variable seed; output: 1 byte
- Weaknesses
  - Bias
    - Pr[2nd byte = 0] = 2/256 (twice as random)
      - Other bytes are biased too (e.g., 1st,3rd)
      - It is recommended that the first 256 byes are ignored
    - Pr[00] = $1/256^2 + 1/256^3$
      - Bias starts after several gigabytes but it is still a distinguisher
  - Related keys
- It is recommended not to use RC4 but modern CSPRNG

Linear Feedback Shift Register

UNIVERSITÀ DI PISA

- p_i = feedback coefficient (If p_i == 1, the feedback is active; otherwise it is not)



$$s_m \equiv p_{m-1}s_{m-1} + \cdots + p_1 s_1 + p_0 s_0 \bmod 2$$

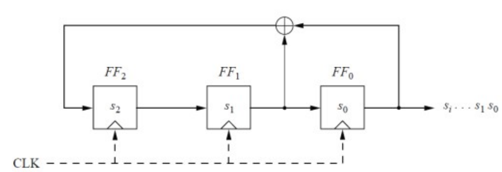
$$s_{m+1} \equiv p_{m-1}s_m + \cdots + p_1 s_2 + p_0 s_1 \bmod 2$$

$$s_{i+m} \equiv \sum_{j=0}^{m-1} p_j \cdot s_{i+j} \bmod 2, s_i, p_j \in \{0,1\}, i = 0, 1, 2, \cdots$$

LFSR is periodical

UNIVERSITÀ DI PISA

- LFSR
  - Degree: 3
- Sequence of states



clk	FF_2	FF_1	$FF_0 = s_i$
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

← The initial state (*seed*)

vettore coefficienti è stretto per

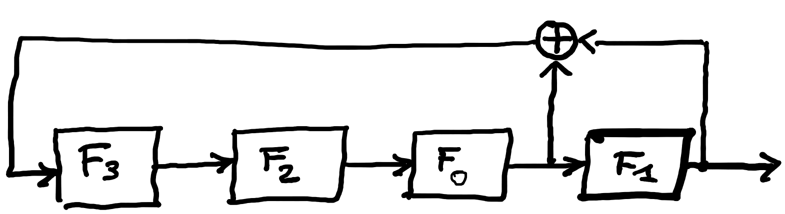
← The sequence of states is *periodical*

LFSR - Properties

- Properties
  - Seed = initial state of the register
    - All 0's state must be avoided
  - Degree = number of storage units = NUMERO DI FLIP-FLOP
    - Degree = 8
  - Periodic
- Maximum-length LSFR
  - Theorem
    - The maximum sequence length generated by an LFSR of degree m is $2^m - 1$
  - Maximum-length LSFR can be easily found

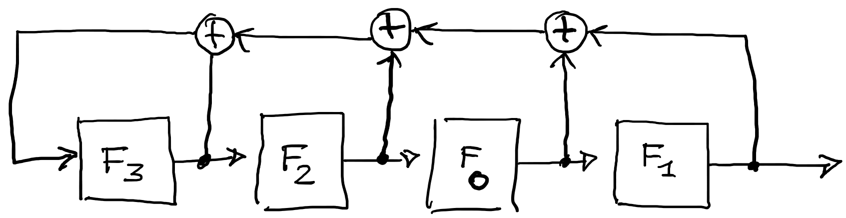
LFSR – example #1

- LFSR with maximum output sequence
  - Degree m = 4
  - Coefficients: $p_3 = 0$, $p_2 = 0$, $p_1 = 1$, $p_0 = 0$
  - Period = $2^m - 1 = 15$



LFSR – example #2

- LFSR with non-maximum output sequence
  - Degree m = 4
  - Coefficients: $p_3 = 1$, $p_2 = 1$, $p_1 = 1$, $p_0 = 1$
  - Period = 5



LFSRs are not good for crypto

- Pros:
  – LFSRs have good statistical properties
- Cons
  – Periodical
  – Linear

LFSRs are not good for crypto

- Known-Plaintext attack against LFSR
  1. Given 2m pairs (pt, ct), the adversary determines a prefix of the sequence s_i
  2. Then, the adversary determines *feedback coefficients* by solving a system of m linear equations in m unknowns
  3. Finally, the adversary can "build" the LFSR and produce the entire sequence

LSFRs are not good for crypto

- Have LSFRs to be thrown away?
  - Use a non-linear combination of several LFSRs to build strong cryptosystems
    - E.g., use AND
  - E.g.: Trivium (2003)

State of the art

- Software-oriented
  - RC4 and SEAL
    - Very well-investigated; secure
- Hardware-oriented
  - LFSR-based
    - Many have been broken
  - GSM A5/1 and A5/2
    - A5/1 used to be secret but was reverse-engineered
    - A5/2 has serious flaws
    - Neither of them is recommended nowadays
    - A5/3 (KASUMI) is used but it is a block cipher

State of the art

- eSTREAM Project
  - ECRYPT NoE
    - Call for stream ciphers; 34 candidates
  - Profile 1. Stream ciphers for software applications with high throughput requirements
    - HC-128, Rabbit, Salsa20/12, SOSEMANUK
  - Profile 2. Stream ciphers for hardware applications with restricted resources
    - Grain v1, MICKEY v2, Trivium

eSTREAM performance

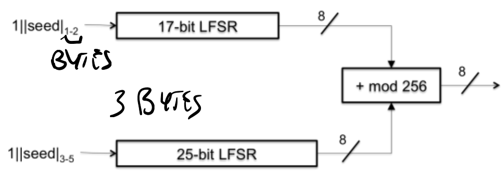
- RC4 126 Mb/s (*)
- Salsa 20/12 643 Mb/s
- Sosemanuk 727 Mb/s
- (*) AMD Opteron 2.2. GHz (Linux)

Stream Ciphers

CONTENT SCRAMBLING SYSTEM (CSS)

Content Scrambling System

- Seed (key)
  - initial states of the LFSRs 5 bytes (~~80~~ 40 bit)
- Each round
  - 8 CLK cycles
  - Each LFSR produces 8 bits
  - LFSR's outputs are added mod 256(*) so producing the key stream
    - (*) neglect carry bit for simplicity



TUTTE LE POSSIBILI CHIAVI 2^{40}

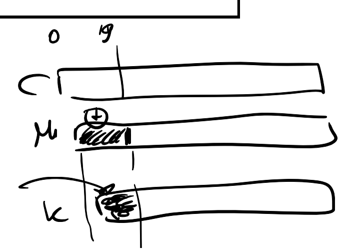
38

Content Scrambling System

- Easy to break in 2^{17} steps ($<< 2^{40}$)
- Known-plaintext attack
  - A prefix$|_{1\text{-}20}$ of the (cleartext) movie is known => a prefix of the keystream$|_{1\text{-}20}$ can be computed
    - E.g., 20 initial bytes in mpeg
- For details
  - https://www.cs.cmu.edu/~dst/DeCSS/Kesden/

39

SE CONOSCO IL PREFISSO (20 BYTE) DEL TESTO CIFRATO POSSO CONOSCERE 20 BYTE DELLA CHIAVE



Content Scrambling System

- Attack algorithm
  - For all possible initial setting of LFSR-17 (2^{17})
    1. Run LFSR-17 to get 20 bytes of output
    2. Subtract LFSR-17$|_{1\text{-}20}$ from keystream$|_{1\text{-}20}$ and obtain a candidate output of LFSR-25$|_{1\text{-}20}$
    3. Check whether LFSR-25$|_{1\text{-}20}$ is consistent with LSFR-25
       a. If it is consistent then we have found correct initial setting of both and the algorithm is finished!
       b. Otherwise, go to 1 and test the next LFSR-17 initial setting
  - Using key, generate entire CSS output
  - Complexity
    - At most, the attack need to try all the possible initial setting of LFSR-17 (2^{17})

WORST CASE