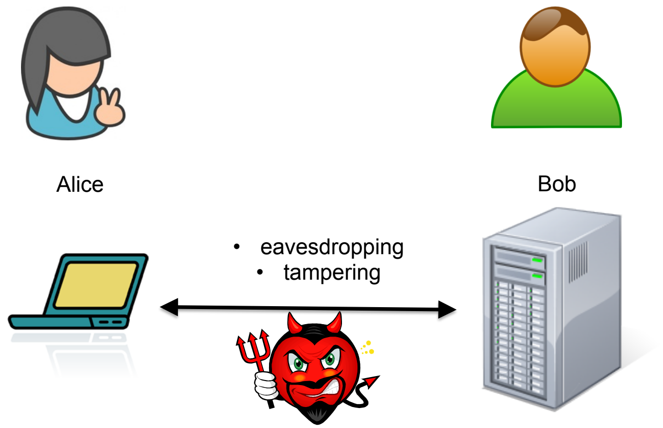
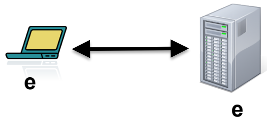
Symmetric Encryption

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
g.dini@iet.unipi.it

Secure communication

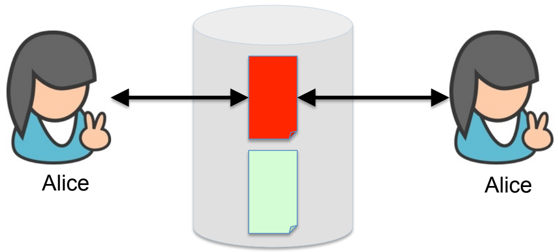


SSL



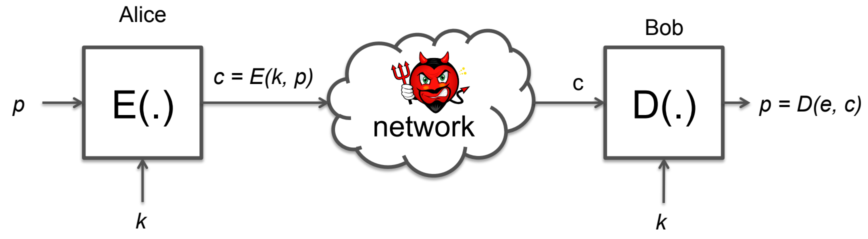
- **Handshake protocol**: establish a **shared secret key** by means of public key cryptography
  (2nd part of the course)
- **Record protocols**: use **shared secret key** to transmit data
  - Ensure confidentiality and integrity
  (1st part of the course)

Encrypted files



- Analogous to secure communication
  - Alice todays sends a message to Alice tomorrow

Building block: symmetric encryption



- *E*, *D*: cipher *k*: **shared secret key** (128 bits)
- *p*, *c*: plaintext, ciphertext

- Encryption algorithm is **publicly known**
  - Never use proprietary algorithm

Kerchoff's principle (19th century)

- *A cryptosystem should be secure even if everything about the system, except the key, is public knowledge*
- *The enemy knows the system* (Shannon's maxim)
- Pros: maintaining security is easier
  - Keys are small keys
    - Keeping small secrets it's easier than keeping large secrets
    - Replacing small secrets, once possibly compromised, is easier than replacing large secrets

Security through Obscurity

- StO attempts to use secrecy of design or implementation to provide security
- History shows that StO doesn't work
  - GSM/A1 disclosed by by mistake
  - RC4 disclosed deliberately
  - Enigma disclosed by intelligence
  - ... many others…
- Solely relaying on StO is a poor design decision
  - A secondary measure: defense in depth

Things to remember

- Cryptography is
  - a very useful tool
  - the basis for many mechanisms
- Cryptography is not
  - The solution to all security problems
  - Reliable unless implemented and used properly
  - Something you should try to invent yourself

Cipher definition

- **(DEF)** A cipher defined over ($\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$) is a pair of "efficient" algs (E, D) where

$$E: \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C} \qquad D: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P}$$

- s.t.

$$\forall p \in \mathcal{P}, k \in \mathcal{K} : D(k, E(k,p)) = p$$

Consistency equation

  – *E* may be randomized; *D* is always deterministic

Security of a cipher

- (**Informal def**) A symmetric cipher is secure iff for each pair (*p*, *c*) then
- given *c*, it is "difficult" to determine p without knowing e, and vice versa
- given *c* and *p*, it is difficult to determine *e*, unless it is used just once

What's a secure cipher

- Attacker ability: **cipher-text only**
- Possible security requirements
  – Attacker cannot recover secret key
  – Attacker cannot recover plaintext
- Shannon's idea
  – **Cipher-text should not reveal any information about plaint-text**

Perfect secrecy
(Shannon, 1949)

- A cipher (*E*, *D*) defined over ($\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$) has **perfect secrecy** iff

$$\forall p \in \mathcal{P}, c \in \mathcal{C} : \Pr\left(P = p \mid C = c\right) = \Pr(P = p)$$

where P is a random variable in $\mathcal{P}$ and C is a random variable in $\mathcal{C}$

Information theoretical secure cipher

Unconditionally secure cipher

Shannon's Theorem

UNIVERSITÀ DI PISA

- **Theorem**. In a perfect cipher $|\mathcal{K}| \geq |\mathcal{P}|$, i.e., the number of keys cannot be smaller than the number of messages
  - **Proof**. By contradiction.
- A perfect cipher is impractical!

Unconditional security

UNIVERSITÀ DI PISA

- Perfect secrecy = unconditional security
  - An adversary is assumed to have infinite computing resources
  - Observation of the CT provides no information whatsoever to the adversary
- Necessary condition is that
  - the key bits are truly randomly chosen and
  - key len is at least as long as the msg len

Perfect secrecy
(another definition)

UNIVERSITÀ DI PISA

- **Definition**. A cipher (E, D) over $(\mathcal{K}, \mathcal{P}, \mathcal{C})$ has perfect secrecy iff $\forall m_1, m_2 \in \mathcal{P}$ $(|m_1| = |m_2|)$, $\forall c \in \mathcal{C}$, $\Pr(E(k, m_1) = c) = \Pr(E(k, m_2) = c)$, where $k \xleftarrow{random} \mathcal{K}$

One Time Pad
(Vernam cipher, 1917)

UNIVERSITÀ DI PISA

- Let m be a t-bit message, i.e., $m \in \{0,1\}^t$
- Let k be a t-bit key stream, $k \in \{0, 1\}^t$, where each bit is truly random chosen
- *Encryption:* $E(k, m) = m \oplus k$
- *Decryption:* $D(k, c) = c \oplus k$
- Very fast enc/dec

OTP has perfect secrecy

- **Theorem**. OTP has perfect secrecy iff

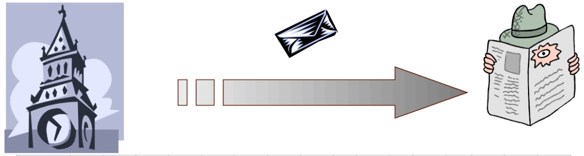
  1. $\forall m, m' \in \mathcal{P} : len(m) = len(m')$
  2. $\forall m \in \mathcal{P} : \Pr(M = m) \neq 0$
  3. $k \xleftarrow{r} \mathcal{K}$

- OTP uses a minimal number of keys (minimality)

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



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

Property of XOR

- The following theorem explains why $\oplus$ is so frequently used in cryptography.
- **Theorem**. Let Y be a random variable on $\{0, 1\}^n$, and X an ***independent uniform*** variable on $\{0,1\}^n$. Then $Z = Y \oplus X$ is uniform on $\{0,1\}^n$.
- **Proof**. (for $n = 1$)

Properties of OTP
(pros and cons)

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***

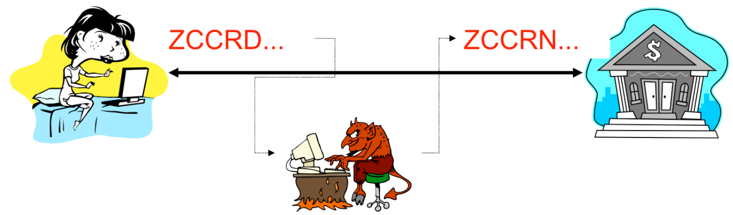
Properties of OTP
(pros and cons)

CONS

- **Long keys**
  – Key len >= msg len
- **Keys must be used once: avoid two time pad!**
  – C1 = M1 xor K, C2 = M2 xor K =>
  – C1 xor C2 = M1 xor M2 => M1, M2 (due to redundancy of English and ASCII)
- **A Known-PlainText attack breaks OTP**
  – Given (m, c) => k = m xor c
- **OTP does not provide integrity, even worse *OTP is malleable***
  – Modifications to cipher-text are undetected and have predictable impact on plain-text

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

On malleability

- Alice sends Bob: $c = p \oplus e$
- The adversary intercepts c and transmits Bob $c' = c \oplus r$ (perturbation)
- Bob receives c' and obtains $p' = p \oplus r$
  – **The modification goes undetected**
  – **Predictable impact on the plaintext**

Making OTP practical (1/3)

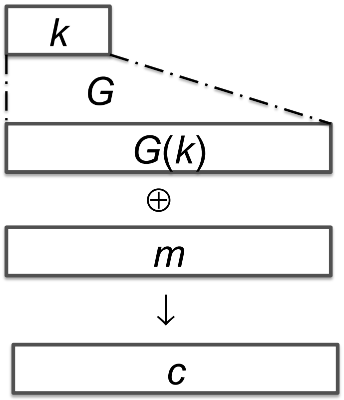
- **Idea:** replace random key by pseudo-random key
- Pseudo-Random Generator G is an ***efficient*** and ***deterministic*** function

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

Seed space **Key-stream space**

Making OTP practical (2/3)

$$c = E(k,m) = G(k) \oplus p$$
$$m = D(k,c) = G(k) \oplus c$$



Making OTP practical (3/3)

- **Can OTP now have perfect secrecy?**
  - We need a new definition of security!
- **Security will depend on the specific PRG**
  - PRG must be/appear unpredictable
  - PRG must look random, i.e., indistinguishable from a TRG for a limited adversary
  - There **exist no efficient algorithm** to distinguish PRNG output from a TRG output
  - CSPRNG

Computational security, a new definition of security

Statistical tests

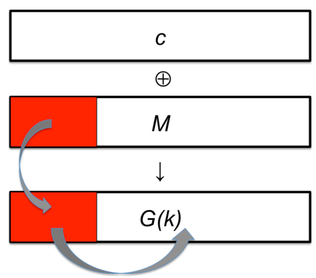
- Measure the quality of a random bit generator
  - Probabilistically determine whether sample output sequences possess certain attributes that a truly random sequence would be likely to exhibit
    - Ex.: a sequence should roughly have the same number of 1's as 0's
  - A generator may be **rejected** or **accepted** (= not rejected)
- Provide **necessary conditions** only

PRG must be unpredictable

- (**Informal DEF**) PRG is **predictable** if $\exists\, i$ and an **efficient** algorithm A s.t.

$$A\left(G(k)\big|_{0,\cdots,i}\right) \rightarrow G(k)\big|_{i+1,\cdots,n-1}$$

- Then OTP is not secure!
- Even $A\left(G(k)\big|_{0,\cdots,i}\right) \rightarrow G(k)\big|_{i+1}$ is a problem



Building PRG is hard

- "Random numbers should not be generated with a method chosen at random." —Donald E. Knuth
- "The generation of random numbers is too important to be left to chance." —Robert R. Coveyou
- "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin" —John von Neumann

Weak PRG
(don't use in crypto!)

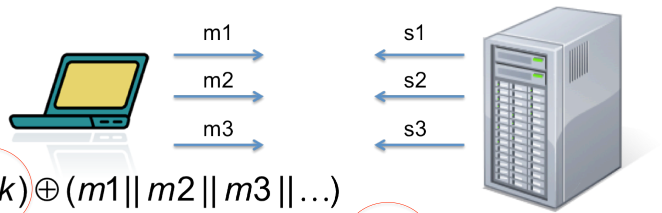
- **Linear congruential generator (LCG)**

$$r[0] = seed$$
$$r[i] = a \cdot r[i-1] + b \mod p$$

- glibc random() is similar to LCG
  - **Good statistics but predictable!**

Real world examples

- MS-PPTP



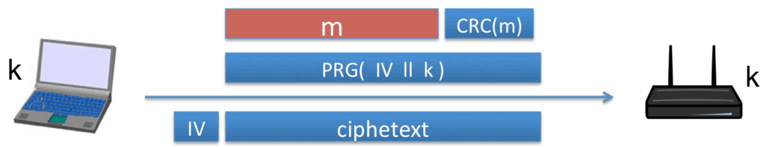
$$G(k) \oplus (m1 \,||\, m2 \,||\, m3 \,||\, \ldots)$$

$$G(k) \oplus (s1 \,||\, s2 \,||\, s3 \,||\, \ldots)$$

Two time pad!

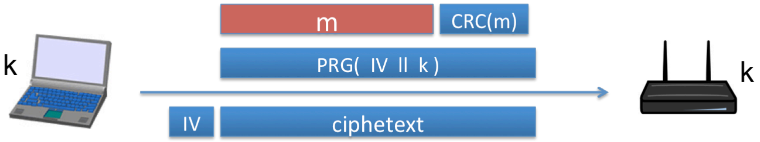
$K = (K_{c \to s}, K_{s \to c})$ is the right way!

Real world examples: 802.11b WEP



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

Real world examples: 802.11b WEP

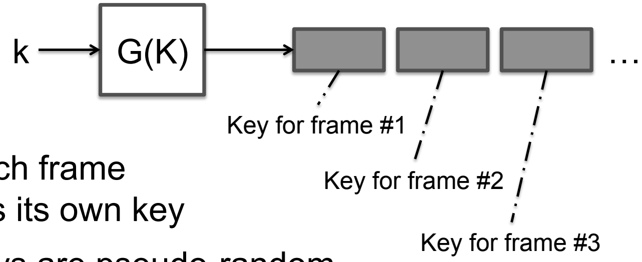


Key for frame #1: 1||k
Key for frame #2: 2||k
Key for frame #3: 3||k
…

- Related keys
- FMS 2001 attack can recover K in 10^6 frames (now 40 Kframes)
- **Avoid related keys!**

Real world examples: 802.11b: WEP

- **A better construction**



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

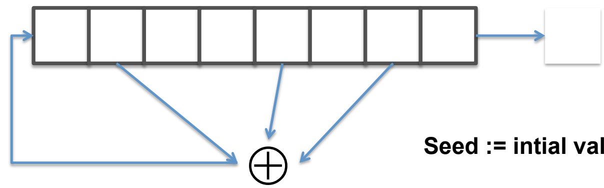
Real world examples: old examples

- **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**

Real world examples

Old example (hw): CSS badly broken!

- **Linear Feedback Shift Register (LFSR)**



- **Seed := intial value**

- DVD encryption (CSS): 2 LFSRs
- GMS encryption (A5/1,2): 3 LFSR
- Bluetooth (E0): 4 LFSRs

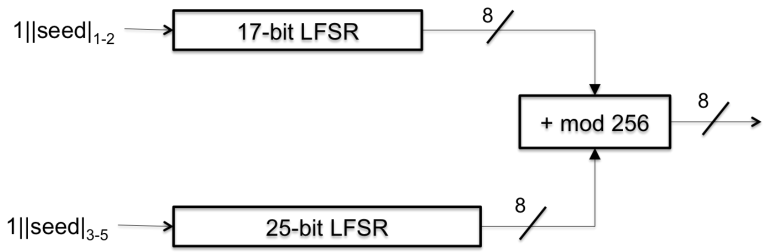
All broken!

LSFRs: not a good idea for crypto

- LSFRs are a bad choice in cryptography
- LSFRs have pros that are cons in crypto
  - They are periodical
    - A LSFR-m as at most a 2^m-1 period
  - They are linear
    - A LSFR-m can be expressed as a *m*-degree polynomial
    - As soon as we know *m* outputs of, we can efficiently compute the polynomial's coefficient by solving a system of linear equations
    - Outputs can be computed from a KPT attack
- Have LSFRs to be thrown away?
  - No, provided you use non-linear combinations (e.g. AND) of LSFRs
  - Trivium stream cipher (2003)

Real world examples

- **CSS**$^{(*)}$: seed = 5 bytes (the key)

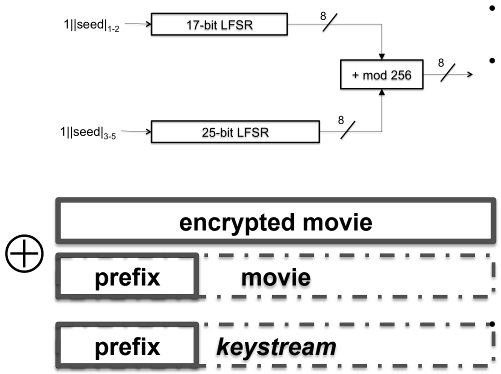


- Easy to break in time 2^{17}

(*) More can be found here https://www.cs.cmu.edu/~dst/DeCSS/Kesden/

Real world examples

- **CSS**



- A prefix of the movie is known (e.g., 20 bytes in mpeg):**KPT!**
- Then a prefix of CSS$|_{1\text{-}20}$ can be computed
- For all possible initial setting of LFSR-17 (2^{17})
  - Run LFSR-17 to get 20 bytes of output
  - Subtract from CSS prefix → candidate 20 bytes output of LFSR-25
  - If consistent with LFSR-25 → found correct initial setting of both!!
- Using key, generate entire CSS output

Modern stream ciphers (eStream)

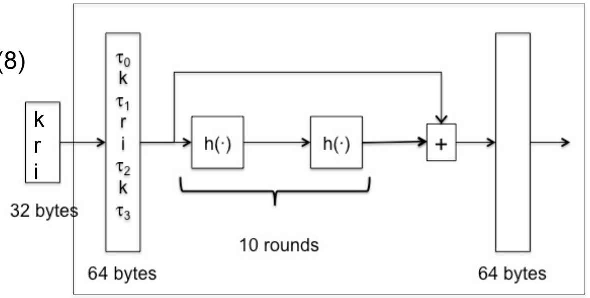
- PRG: $\{0,1\}^s \times R \to \{0,1\}^n$, $n \gg s$

seed nonce keystream

- **nonce**: a non-repeating value for a given key
- The pair (*seed*, *nonce*) is never used more than once
  - You can reuse the key because the nonce makes (k, r) unique

eStream project: Salsa 20

- Salsa20: $\{0,1\}^s \times \{0,1\}^{64} \rightarrow \{0,1\}^n$, s = 128, 256
- Salsa20: H(k, (r, 0)) || H(k, (k, 1)) || …
- h(): invertible function, to be fast on x86
- τ_x: constant (8),
  i : index/counter (8)
  k: (16)
  r: (8)
- **No significant attacks!**



Performance

AMD Opteron 2.2 GHz (Linux)

	PRG	**Speed (Mb/s)**
	RC4	126
eStream	Salsa 20/12	643
	Sosemanuk	727

TRG

- RBG requires a naturally occurring source of randomness

RBG

Sequence of **statistically independent** and **unbiased** bits

Probability of emitting a bit (1 or 0) value does not depend on the previous bits

Probability of emitting a bit value (1 or 0) is equal to 0.5

HW-based TRG

- **HW-based RBGs exploit the randomness in some physical phenomena**
  - elapsed time between emission of particles during radioactive decay
  - thermal noise from a semiconductor diode or resistor
  - the frequency instability of a free running oscillator
  - the amount a metal-insulator semiconductor capacity is charged during a fixed period of time
  - air turbulence within a sealed disk drive which causes random fluctuations in disk drive sector read latency times
  - sound from a microphone or video from a camera

SW-based TRG

UNIVERSITÀ DI PISA

- **Random processes used by SW-based RBGs include**
  - The system clock
  - Elapsed time between keystrokes or mouse movement
  - Content of input/output buffers
  - User input
  - Operating system values such as system load and network statistics
  - A well-designed SW-based RBG uses as many sources as available

TRG

UNIVERSITÀ DI PISA

- TRG must not be subject to observation and manipulation by an adversary
- The natural source of randomness is subject to influence by external factors and to malfunction
- TRG must be tested periodically

RBG Test Suites

UNIVERSITÀ DI PISA

- Diehard Battery of Tests of Randomness CD, 1995. **http://www.stat.fsu.edu/pub/diehard/**
- NIST test suite for random numbers **http://csrc.nist.gov/groups/ST/toolkit/rng/index.html**

Symmetric Encryption

BLOCK CIPHERS

Block cipher

- Block ciphers break up the plaintext in blocks of fixed length *n* bits and encrypt one block at time

- E: $\{0,1\}^n \rightarrow \{0,1\}^n$ D: $\{0,1\}^n \rightarrow \{0,1\}^n$
- ***E* is a permutation** (one-to-one, invertible)

Canonical example

Block ciphers

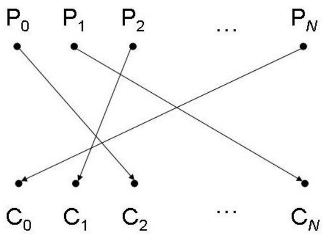
- DES n = 64 bits, k = 56 bits
- 3DES n = 64 bits, k = 168 bits
- AES n = 128 bits k = 128, 192, 256 bits

Performance (AMD Opteron, 2.2 GHz)

- RC4 126 MB/s
- Salsa20/12 643 MB/s
- Sosemanuk 727 MB/s
- 3DES 13 MB/s
- AES-128 109 MB/s

True Random Cipher

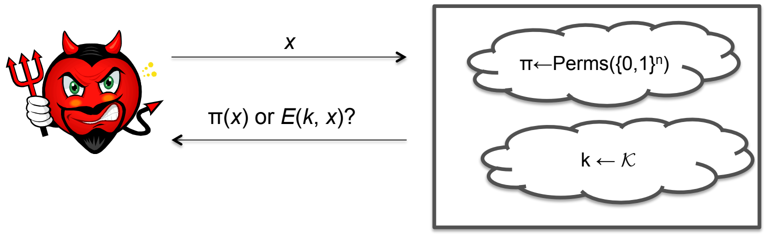
$N = 2^n$



- A true random cipher is perfect
- Implement all possible permutations: $2^n!$ permutations
- A random key for each permutation
- Key size := $\log_2 2^n! \approx (n - 1.44)\, 2^n$
  exp in the block size!

Practical block cipher

- In practice, the encryption function corresponding to a randomly chosen key should appear as a randomly chosen permutation



Exhaustive key search

- **Problem**. Given a few pairs (p_i, $c_i = E(e, p_i)$) find e
  - ***Known-plaintex attack***
- **Theorem**. Given $\lceil (k+4)/n \rceil$ pairs of plaintext ciphertext, a key can be recovered by exhaustive key search in an expected time $O(2^{k-1})$

Exhaustive key search

- **DES**, $\forall p, c$, there is at most **one** key e s.t. c = DES(e, p) with probability ≥ 1 − 1/256 = 99.5% (**unicity probability**)
- **DES**, for two pairs (p_1, c_1 = DES(e, p_1)), (p_2, c_2 = DES(e, p_2)), unicity probability ≈ $1 - 1/2^{71}$
- **AES-128**, given two input/output pairs, unicity probability ≈ $1 - 1/2^{128}$
- **⇒ two input/output pairs are enough for exhaustive key search**

DES challenge

```
p = "The unknown messages is: XXX …"
          c1      c2      c3
```

- **Goal**. Find $e \in \{0,1\}^{56}$ s.t. c_i = DES(e, p_i), i = 1, 2, 3
- 1997: Internet search – 3 months
- 1998: EFF machine (deep crack) – 3 days (250K$)
- 1999: combined search – 22 hours
- 2006: COPACABANA (120 FPGAs) – 7 days (10K$)
- **⇒ 56-bit ciphers should not be used**

Triple DES (3DES)

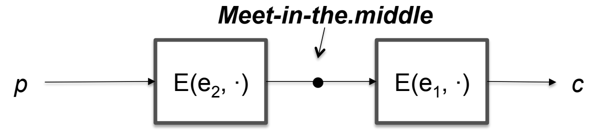
- **EDE –** 3E((e_1, e_2, e_3), p) = $E(e_1, D(e_2, E(e_3, p)))$
  - $e_1 = e_2 = e_3$ ⇒ 3DES → DES (backward compatibility)
  - Key size = 168-bits
  - 3 times slower than DES
  - Simple attack ≈ 2^{118}
  - Standard (ANSI X9.17 and ISO 8732)

Two-times DES (2DES)

- c = $2E((e1, e2), m) = E(e2, E(e1, m))$
  - Key size: 112 bits
  - 2 times slower than E
- Completely unsecure
  - Naïve approach: 2^{2k}
  - Meet-in-the-middle attack
    - Time complexity: 2^{56} (doable nowadays!)
    - Space complexity: 2^{56} (lot of space!)
    - Expected number of false positives: 2^{2k-tn}
- 2E brings no advantage

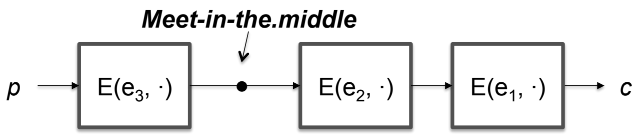
Meet-in-the-middle attack

- **Intuition**: given (c, p), find (e_1, e_2) s.t. $E(e_1, E(e_2, p)) = c$
  - Time complexity < 2^{63} (doable nowadays!)
  - Space complexity = 2^{56} (lot of space!)
  - No advantage in 2E!



Meet-in-the-middle attack

- **3DES**
- Meet-in-the-middle-attack
  - Time = 2^{112} (undoable!)
  - Space = 2^{56} (lot of space!)



Computational security

- A cipher is **computationally (practically) secure** if the ***perceived level of computation*** required to defeat it, using the ***best attack known***, exceeds, by a comfortable margin, the *computation resources of the hypothesized adversary*
  - Now, the adversary is assumed to have a limited computation power

Attack Complexity

- **Attack complexity is the dominant of**:
- **data complexity** — expected number of input data units required
  - Ex.: exhaustive data analysis is $O(2^n)$
- **storage complexity** — expected number of storage units required
- **processing complexity** — expected number of operations required to processing input data and/or fill storage with data
  - Ex.: exhaustive key search is $O(2^k)$

Computational security vs attack complexity

- **A block cipher is computationally secure if**
- **Block size *n* is sufficiently large** to preclude exhaustive data analysis, and
- **Key size *k* is sufficiently** large to preclude exhaustive key search, and
- **No known attack has data and processing complexity significantly less than, respectively, 2^n and 2^k**

Types of attacks

- Attacks are classified according to what information an adversary has access to
  - **ciphertext-only attack (the least strong)**
  - **known-plaintext attack**
  - **chosen-plaintext attack (the strongest)**
- **Fact**. A cipher secure against chosen-plaintext attacks is also secure against CT-only and known-PT attacks
- **Best practice**. It is customary to use ciphers resistant to a chosen-PT attack even when mounting that attack is not practically feasible

Cryptoanalysis
An historical example

Mono-alphabetic substitution

Cleartext alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Key	J	U	L	I	S	C	A	E	R	T	V	W	X	Y	Z	B	D	F	G	H	K	M	N	O	P	Q

P = "TWO HOUSEHOLDS, BOTH ALIKE IN DIGNITY,
IN FAIR VERONA, WHERE WE LAY OUR SCENE"

("Romeo and Juliet", Shakespeare)

P' = "TWOHO USEHO LDSBO THALI KEIND IGNIT
YINFA IRVER ONAWH EREWE LAYOU RSCEN E"

C = "HNZEZ KGSEZ WIGUZ HEJWR VSRYI RAYRH
PRYCJ RFMSF ZYJNE SFSNS WJPZK FGLSY S"

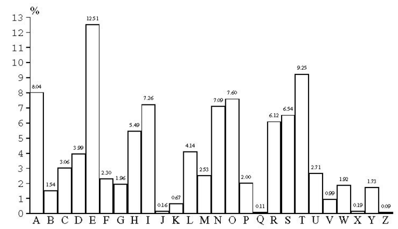
Cryptoanalysis
An historical example

Mono-alphabetic substitution

- The key is a permutation of the alphabet
- **Encryption algorithm**: every cleartext character having position *p* in the alphabet is substituted by the character having the same position *p* in the key
- **Decryption algorithm**: every ciphertext character having position *p* in the key is substituted by the character having the same position *p* in the cleartext
- Number of keys = $26! - 1 \simeq 4 \times 10^{26}$ (number of seconds since universe birth!)

Cryptonalysis
An historical example

- The monoalphabetic-substitution cipher maintains the redundancy that is present in the cleartext
- It can be "easily" crypto-analized with a ciphertext-only attack based on language statistics



Frequency of single characters in English text

Cryptoanalysis: lesson learned

- Good ciphers should hide statistical properties of the encrypted plaintext
- The cyphertext symbols should appear to be random
- A large key space alone is not sufficient for strong encryption function (necessary condition)

Crypto-analysis of DES

attack method	data complexity		storage complexity	processing complexity
	known	chosen		
exhaustive precomputation	—	1	2^{56}	1*
exhaustive search	1	—	negligible	2^{55}
linear cryptanalysis	2^{43} (85%)	—	for texts	2^{43}
	2^{38} (10%)	—	for texts	2^{50}
differential cryptanalysis	—	2^{47}	for texts	2^{47}
	2^{55}	—	for texts	2^{55}

* Table lookup
%: probability of success

LC is the best known analytical attack but it is considered "unpractical"

Cryptanalysis

- **Cryptanalysis** is the ***science*** and, sometimes, the ***art*** of breaking cryptosystems
  - **Classical cryptanalysis**: recovering PT, or even the key, from CT
    - Brute-force attack
    - Analytical attack
  - **Implementation attack**
    - Side-channel analysis (time-, power-analysis)
    - Buffer-overflow
  - **Social Engineering Attacks**
    - Bribing, blackmailing, tricking

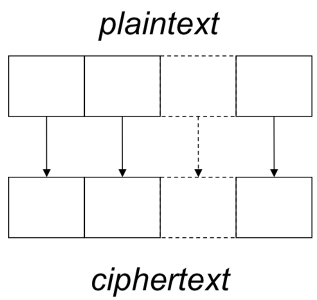
Symmetric Encryption

ENCRYPTION MODES

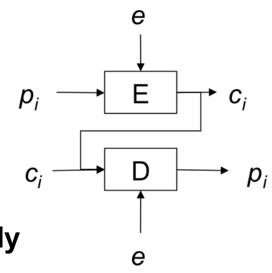
Encryption Modes

- A block cipher encrypts PT in fixed-size *n*-bit blocks
- When the PT len exceeds n bits, there are several modes to the block cipher
  - Electronic Codebook (ECB)
  - Cipher-block Chaining (CBC)
  - Cipher-feedback (CFB)
  - Output feedback (OFB)

Electronic codebook



$$\forall 1 \le i \le t, c_i \leftarrow E(e, p_i)$$
$$\forall 1 \le i \le t, p_i \leftarrow D(e, c_i)$$



PT blocks are encrypted separately

ECB - properties

- **PROS**
  - No block synchronization is required
  - No error propagation
    - *One or moe bits in a single CT block affects decipherment of that block only*
  - Can be parallelized
- **CONS**
  - Identical PT results in identical CT
    - *ECB doesn't hide data pattern*
    - *ECB allows traffic analysis*
  - Blocks are encrypted separately
    - *ECB allows block re-ordering and substitution*

ECB – block replay attack

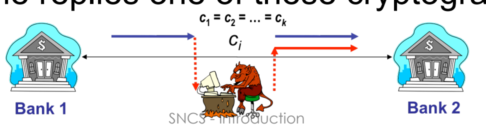
- Bank transaction that transfers a client U's amount of money D from bank B1 to bank B2
  - Bank B1 debits D to U
  - Bank B1 sends the "credit D to U" message to bank B2
  - Upon receiving the message, Bank B2 credits D to U
- Credit message format
  - Src bank: M (12 byte)
  - Rcv banck: R (12 byte)
  - Client: C (48 byte)
  - Bank account: N (16 byte)
  - Amount of money: D (8 byte)
- Cipher: n = 64 bit; ECB mode

ECB – block replay attack

- Mr. Lou Cipher is a client of the banks and wants to make a fraud
- Attack aim
  - To replay Bank B1's message "credit 100$ to Lou Cipher" many times
- Attack strategy
  - Lou Cipher activates multiple transfers of 100$ so that multiple messages "credit 100$ to Lou Cipher" are sent from B1 to B2
  - The adversary identifies at least one of these messages
  - The adversary replies the message several times

ECB – block replay attack

- Mr. Lou Cipher performs *k* equal transfers
  - credit 100$ to Lou Cipher → c1
  - credit 100$ to Lou Cipher → c2
  - ...
  - credit 100$ to Lou Cipher → ck
- Then, he searches "his own" CT in the network
  - k equal CTs!
- Finally he replies one of these cryptograms



ECB – block replay attack

UNIVERSITÀ DI PISA

- An 8-byte timestamp field T (block #1) is added to the message to prevent replay attacks

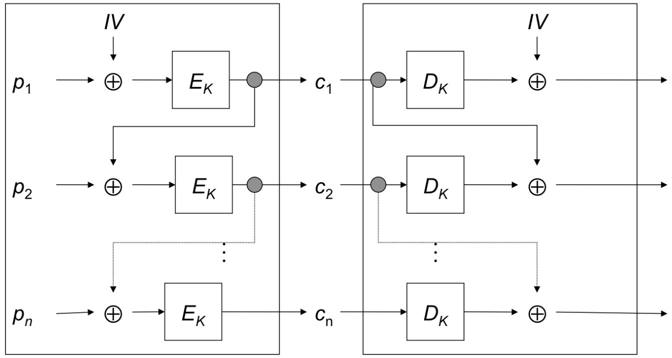
block no.	1	2	3	4	5	6	7	8	9	10	11	12	13
	T	M	M	R	C	C	C	C	C	C	N	N	D

- However, Mr Lou Cipher can
  - Identify "his own" CT by inspecting blocks #2-#13
  - Intercept any "fresh" CT
  - Substitute block #1 of "his own" CT with block #1 of the intercepted "fresh" block
  - Replay the resulting CT

Cipher block chaining (CBC)

UNIVERSITÀ DI PISA

$$c_0 \leftarrow IV. \forall 1 \le i \le t, c_i \leftarrow E_k(p_i \oplus c_{i-1})$$
$$c_0 \leftarrow IV. \forall 1 \le i \le t, p_i \leftarrow c_{i-1} \oplus D_k(c_i)$$



CBC - properties

UNIVERSITÀ DI PISA

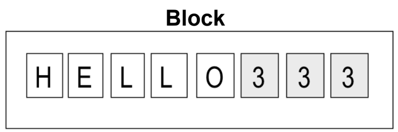
- Chaining dependencies: c_i depends on p_i and all preceding PT blocks
- Encryption is *randomized* by using IV
  - CBC is *non deterministic*
    - Identical ciphertext results from the same PT under the same key and IV
  - IV is a *nonce*
- CT-block reordering affects decryption
- IV can be sent in the clear but its integrity must be guaranteed
- CBC suffers from Error propagation
  - Bit errors in c_i affect decryption of c_i and c_{i+1} (*error propagation*)
  - CBC is self-synchronizing (*error recovery*)
  - CBC does not tolerate "lost" bits (*framing errors*)

Padding – the PKCS#5 standard

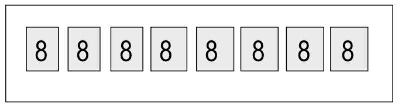
UNIVERSITÀ DI PISA

- Padding is necessary when PT len is not a block multiple

If PT len is NOT a block multiple
Padding bytes ← #bytes to complete a block



If PT is a block multiple
Padding = block
Each padding byte ← 8



Other encryption modes

- Other encryption modes
  - Cipher Feedback mode (CFB)
  - Output Feedback mode (OFB)
  - Counter mode (CTR)
  - Galois Counter mode (GCM)
  - and many others (e.g., CCM, CTS, …)
- In CFB, OFB, CTR a block cipher is used as stream cipher / pseudo-random generator
- In GCM a block cipher guarantees confidentiality and authentication and integrity
- Block ciphers are very versatile components