Crittografia = metodi × elaborare × memorizzare × trasmettere informazioni in presenza di agenti ostili

+

Crittoanalisi = analisi

=

Crittologia

TENTATIVO

One Time Pad

R = M = C

✓ sicuro

✓ De E veloci

× Key lunghe $|K| \geq |M|$

! difficile da usare

* malleabile

→ **Cifrari a flusso**

× perfettamente sicuro $|k| \neq |m|$

Key ~~randomica~~ → key PSEUDORANDOMICA

Pseudo Random Generator PRG

funzione DETERMINISTICA
COMPUTABILE EFFICACEMENTE

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

Seed — key

Sicuro quando è IMPREDICIBILE

$G : K \longrightarrow \{0,1\}^n$ è PREDICIBILE se :
$\exists$ un ALGORITMO EFFICIENTE A e $\exists\ 1 \leq i \leq n-1$ TALE CHE
$P[A(G(k)|_{1,\dots,i}) = G(k)|_{i+1}] > \frac{1}{2} + \varepsilon$ PER UN ε non TRASCURABILE
$k \leftarrow K$

$E(k,m) = G(k) \oplus m = c$ — c è il testo cifrato

$D(k,c) = G(k) \oplus c = m$ — key K è RANDOM e ONE-Time-Key

un PRG è SICURO se $\forall$ test statistico A, $Adv[A,G]$ è vicino a 0 tanto che sia "IRRILEVANTE"

ATTACCO OTP e CIFRARI A FLUSSO

1 usare la key 2+ volte

2 integrità NON assicurata

! OTP è malleabile → con un Man In The Middle → se si verificano modifiche del testo cifrato

- sapendo il contenuto del msg lo può modificare con qualsiasi cosa
- Non sono riconoscibili

RC4 è PRG

salva in un array S la pseudorandom permutazione dei num $0 \dots 255$

```
for i=0 to 255 do:
  S[i] = i
j =0

for i=0 to 255 do:
  k = s[i % |s|]   //extract one byte from seed
  j = (j + S[i] + k) % 256
  swap(S[i], S[j])
```

! il 2° bit sarà per lo più delle volte in chiaro → LA PROBABILITÀ CHE SIA 0 è molto alta

* ATTACCO "chiavi correlate"

Content Scrambling System

$\in$ PRG

array Linear Feedback Shift Register LSFR → seed = stato iniziale

× obsoleto
× poco affidabile

eStream

$\in$ PRG

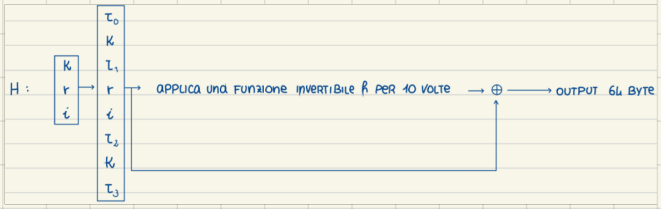
$: \{0,1\}^s \times \underbrace{R}_{\text{nonce}} \longrightarrow \{0,1\}^n$ con $n >> s$

nonce = valore non ripetibile per la stessa key

$E(k,m,r) = m \oplus \text{eStream}(k,r)$

⟶ **salsa 20** $= H(k,(r,0)) \| H(k,(r,1)) \| \cdots$ dove H è def come:

costanti prefissate



test statico su $\{0,1\}^n$ è un alg A | $A(x)$ output 0 v 1 ovvero $A(x): \{0,1\}^n \longrightarrow \{0,1\}$

Advantage

o VANTAGGIO

$$[A,G] = \left| \underset{k \leftarrow k}{P}[A(G(k)) = 1] - \underset{r \leftarrow \{0,1\}^n}{P}[A(r) = 1] \right| \in [0,1]$$

$\in [0,1]$ → A distingue G

A non distingue G da random

Semanticamente Sicuro

Q, se per tutte le A "efficienti", $Adv[A,Q]$ è trascurabile

Thm

G è un PRG SICURO ⇒ lo Stream cipher Q derivato da G è SEMANTICAMENTE SICURO

In particolare $\forall A, \exists$ PRG avversario B (come test statistico) | $Adv_{SS}[A,Q] \leq 2 * Adv_{PRG}[B,G]$

Cifrari a blocchi

STRUTTURA DI FEISTEL → È INVERTIBILE

Fasi uguali in cui metà dei dati da elaborare vengono permutati

in ogni FASE si usa una parte espansa della key originale

DES Algorithm

Blocchi a 64 bit

key a 56 bit

- resistente ad ANALISI CRITTOGRAFICHE DIFFERENZIALI e LINEARI
- ! usa Feistel
- ha effetto VALANGA = un minimo cambiament in INPUT ⟹ un OUTPUT totalmente diverso

* Exhaustive Search

Trudy ha qualche blocco di PT e la loro CT

prova tutte le chiavi possibili sui pochi blocchi che ha [Brute Force] ! IL MIN di BLOCCHI NECESSARI È 3

trova la key

3DES

def 3 key con cui criptare il msg 3 volte

⇒ key da 168 bit

vale anche x 3DES → ma il tempo necessario x la ricerca è >>

~~2DES~~

* Meet in the middle

→ tabella

key	E(key, msg)

x testare D(key, ct)

È nella 2ª colonna della tabella?

SÌ ho trovato la coppia di chiavi

DESX

$EX(k_1, k_2, k_3, msg) = k_1 \oplus E(k_2, msg \oplus k_3)$ 184 bit in tempo 2^{120}

* Linear Attacks

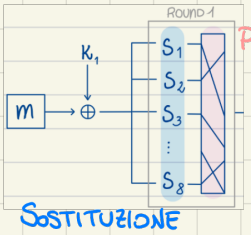
linearità S_5 → tempo di attacco 2^{43}

* Quantum Attacks in tempo $O(|K|^{\frac{1}{2}})$ = facilissimo

AdvanceESstandard ∈ PRP

> iterazioni dipendono dal num di bit della key ORIGINALE

> Decriptazione : con REVERSIBILITÀ ⇒ AES reversibile



PERMUTAZIONE

ogni ROUND è INVERTIBILE

* key recovery con esempi di <PT, CT> della stessa key ⟹ Trovare la key con ricerca esaustiva

* related key attack con esempi di <PT, CT> generato dalla key_i → posso ricavare la key originale sfruttando il meccanismo di espansione

PRFunction

generare valori pseudo-casuali per det. scopi
- key
- msg authentication
- rand numb x protocolli crittografici

INPUT seme iniziale ‖ key

OUTPUT input pseudo-casuale

PRGenerator

genera sequenze di num pseudo-casuali

INPUT seme iniziale ‖ key

OUTPUT sequenza di num che approssima l'aspetto casuale

- PRF - to → PRG

$PRF: K \times \{0,1\}^n \longrightarrow \{0,1\}^n$

$PRG: K \longrightarrow \{0,1\}^{nt}$ t è un parametro arbitrario

$G(k) = F(k, \langle 0 \rangle_n) \| F(k, \langle 1 \rangle_n) \| \cdots \| F(k, \langle t-1 \rangle_n)$ ovvero def G data una key, tramite chiamate della PRF

Dunque se F è SICURO ⟹ G è SICURO. In particolare la key è PARALLELIZZABILE

- PRG - to → PRF

$PRG: K \longrightarrow K^2$

$PRF\ 1bit: K \times \{0,1\} \longrightarrow K$ ovvero $F(k, x \in \{0,1\}) = G(k)[x]$

dato un PRF SICURO, sono sufficienti 3 ROUND con Fiestel per rendere PRP SICURO

$f: k \times \{0,1\}^n \longrightarrow \{0,1\}^n$ PRF sicuro ⟹ $3PRF: k^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ PRP sicuro

Probabilità discreta

× generare key
× alg di cifratura dei dati
} Crittoanalisi è + COMPLESSA 😬

Variabili random è una funzione $X: U \to V$ che induce una distribuzione su V

Variabili random uniforme è una var. rand $r \leftarrow S \mid \forall a \in S: Pr(r=a) = 1/|S|$

Algoritmo random è un alg non deterministico che randomizza l'output

OTP

= è un cifrario che usa ogni key 1 sola volta, ha lunghezza uguale a quella del PT
esegue uno XOR su ogni bit del PT

↳ DECRIPTION
↳ riapplica la key al msg criptato con XOR → ottenendo PT

PRO - veloce nell' ENCRIPTION
CONTRO - key troppo lunghe
- integrità dei dati NON SICURA! → ogni modifica fatta impatta in modo PREVEDIBILE sul msg + modifiche sono RILEVABILI

Sicurezza di Shannon

= un cifrario (E,D) su (R,M,C) ha SICUREZZA PERFETTA se
∀coppia msg∈PT con la stessa lunghezza,
ogni cifrario ha la stessa probabilità di essere generato con la stessa key

Informalmente
Se dal cifrario non è possibile ricavare alcuna informazione sul PT

=> in un cifrario sicuro una chiave k NON può essere + corta della lunghezza del PI

Cifrari a flusso

→ algoritmi crittografici che processano singoli bit/byte alla volta
↓
generando un keystream → = flusso di dati

CON PRG × creare da una key s, una key k molto più lunga → NON SONO SICURAMENTE PERFETTI
↳ × ENCRIPTION/DECRIPTION del msg ← con la stessa procedura di OTP

Esempio OTP

Senza SONO SICURAMENTE PERFETTI

Pseudo Random Generator

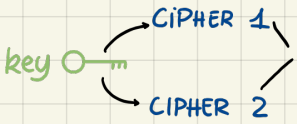
= funzione $G: \{0,1\}^s \to \{0,1\}^n$ con $s << n$
dato seed s, ↦ ritorna una SEQUENZA lunga n >> di s

> COMPUTABILI EFFICIENTEMENTE con algoritmo DETERMINISTICO
> e' SICURO se ∀test statistico A, $Adv_{PRG}[A,G]$ è così vicino allo 0 che sia irrilevante
> dato un $G: k \to \{0,1\}^n$ RICONOSCIAMO che è SICURO se $\forall i \in \{0,\dots,n-1\}$, G è INDISTINGUIBILE in i

genera uno Stream cipher SEMANTICAMENTE SICURO

Attacchi OTP e Stream Ciphers

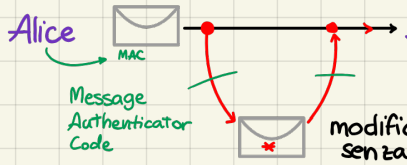
TWO TIME PAD



→ key originale

Soluzione:
Se usassimo una nuova key ∀ SESSIONE nei network
nei dischi è SCONSIGLIATO

NO INTEGRITY



modifica prevedibile senza lasciare traccia

Stream cipher REALI

RC4

key 128 bit PERMUTATA → array S PERMUTATO con indice 1 lineare e indice 2 casuale } effetto Shuffle

* BIAS nell'OUTPUT iniziale

$P[2^{\circ}\text{byte} = 0] = \frac{2}{256}$

eStream

progetto x identificare nuovi Stream cipher

NONCE

valore $<k, r>$ ripetuto per 1 key, usata 1 sola volta

$E(k, m, r) = m \oplus PRG(k, r)$

SALSA 20

key 128/256 bit + NONCE 64 bit → funzione H x 10 volte INVERTIBILE

$n = 2^{73}$ bit max

Sicurezza Semantica

un cipher Q è SEMANTICAMENTE SICURO quando per tutti gli avversari A efficienti,

$Adv_{SS}[A, Q]$ è TRASCURABILE

VANTAGGIO = probabilità che l'esperimento proposto all'avversario ↦ 1 (con INPUT 0) − P[↦ 1 con INPUT 1]

$= \left| P[EXP(0) = 1] - P[EXP(1) = 1] \right|$

GIOCO coppia di esperimenti con bit 0 o 1

Challenger

- sceglie un parametro di sicurezza k ≡ LUNGHEZZA key
- genera una key segreta sk
- genera una coppia di PT m_0 e m_1
- sceglie un bit $b = \{0, 1\}$
  - $b = 0$ → CRITTOGRAFARE m_0
  - $b = 1$ → CRITTOGRAFARE m_1
- invio del testo cifrato a

Avversario

> QUERY su msg ∈ PT ≠ m_1 e ≠ m_0 → ottenendo i corrispondenti CT

- può fare altre QUERY ai CT, ma non al msg CT originale

=> CONGETTURA: Quale msg (m_1 o m_0) è stato CRITTOGRAFATO?

El Gamal

cryptosiste

- poco efficiente — CT lungo |PT| ×2
- sicurezza dipende dal computamento

GENERAZIONE CHIAVI

scelgo p e g radice primitiva di p → scelgo $a = [0, p-2]$ esponente → con $A = g^a \bmod p$

public key = (p, g, A)

private key = a

Block cipher

= cifrario randomizzato basato su

STRUTTURA di FEISTEL

↳ permutare i dati da elaborare

PT diviso in 2 metà DX e SX

una funzione f crittografica

↓

usata sulla metà DX

DX swap SX

× 16/32 volte

NB: Per decriptare si usa la stessa tecnica ma con key

DES

DEX

AES

Puzzle di Merkle

scambio di key tramite un PUZZLE

! Base della crittografia asimmetrica

problemi risolvibili del calcolo

↓

semplice da risolvere per i 2 comunicanti

ma NON per un intruso

Diffie-Hellman

protocollo teorico

key pubblica

* Man in the middle

Trapdoor TDF

$X \to Y$ è una tripla (G, F, F^{-1})

El Gamal

crittosistema randomizzato basato su Diffie-Hellman

- poco efficiente CT lungo $|PT| \times 2$
  - sicurezza dipende dal computamento di log

GENERAZIONE CHIAVI

scelgo p
e
g radice primitiva di p → scelgo $a = [0, p-2]$ esponente → computo $A = g^a \bmod p$ → public key = (p, g, A); private key = a

CRITTAZIONE

Fondamenti

Dataleak

= rilascio di informazioni private

esempi → username & password

→ Fullz (+GRAVI) = informazioni personali

Jamming

& Attacchi Radio

= intasare un canale di comunicazione tanto da non permettere a tutti quelli che vi hanno accesso di non poter ricevere / trasmettere dati

↓

se si usano frequenze portanti => MAX EFFICACIA
↑ ENERGIA da USARE

(↓)
- Frequency
- Hopping
- Spread
- Spectrum

Replay

& Attacchi Radio

= captare per poi reinviare lo stesso segnale

> SEMPLICE

con cancelli automatici o simili

TECNICA Rolling Code verifica tra trasmettitore e ricevente

Sniffing

STRUMENTI

- RICEVITORE (con ANTENNA) alla giusta distanza per la ricezione del segnale

POSSONO essere CIFRATI:

- conoscere la MODULAZIONE usata
- conoscere il PROTOCOLLO usato

Radio Frequency IDentification

ID presente in alcuni oggetti quali tessere, tag o simili contenenti informazioni ulteriori (come nome etc) che possono essere letti tramite contatto

↳ Presenti in un database

Una tipologia molto usata in badge, tessere di mezzi pubblici o chiavette elettroniche per i distributori automatici è EM410X. Una caratteristica fondamentale di questi è che il loro ID non può essere modificato.

↳ ciò si può facilmente raggirare con pseudobadge come proxmark

Standard di comunicazione IEEE 802.11 = WiFi

× creare reti locali interoperabili con reti Ethernet

livello 1 livello fisico di trasmissione → Problema delle Collisioni → lasciare + spazio tra i pacchetti → ? Non Semplice da GESTIRE (IFS)

livello 2 connessione dei dispositivi con MAC address → traffico gestito con CT ← ! Non tutti i messaggi DISASSOCIAZIONE RICEZIONE sono PT

Wired Equivalent Privacy

AUTENTICAZIONE e CONFIDENZIALITÀ DELLE PARTI

modalità di funzionamento →

- **Shared key** CHALENGE & RESPONSE per sapere se il client possiede la key
  - * know plaintext attach
    - → per ricavare la key da tutte le autenticazioni
- **Open System** chi richiede la connessione ha già la key condivisa perchè altrimenti non avrebbe potuto decifrare i pacchetti provenienti dall'access point

* Attacchi Statistici
  - → pacchetti con lo stesso IV
    - ↓
    - se è un IV NOTO ⇒ si può far aumentare il traffico facendo ricircolare pacchetti vecchi

Wifi Protected Access

× rendere + sicuro il protocollo IEEE 802.11 ← SOSTITUENDO WEP

- **PSK** × rete domestica
  - • ogni dispositivo ha una key segreta
- **Enterprise** × reti con molti utenti
- **WPS** autenticazione facilitata + diversi metodi di cifratura
- **TKIP** compatibile con WEP ⇒ DEPRECATO
- **CCMP** basato su AES

KRAK

* Attacco di Replay per reti ~~WPA2~~ ⇒ WPA3
  - → uso di NONCE nella fase di autenticazione
    - ↓
    - che può essere riusato UGUALE × velocizzare le auth. successive ⇒ Si può REINSTALLARE key VECCHIE
      - ↓
      - × risalire alla key di CIFRATURA

* Rough Access Point

- SPOOFING del nome della rete
  +
  invito ad accedere i client ⟶ "PASSWORD ERRATA"
  dato che si trovano su un'altra rete

- E' UN ATTACCO SU WPS/Enterprise
  ↳ [Brute force] o [dizionario] x trovare la password

Privilege Escalation* = possibilità di ottenere + privilegi sul sistema

- installare programmi
- leggere e scrivere su tutti i file privati e nascosti
- modificare impostazioni avanzate di sistema

Access Control List

UNIX: x file NON aperti

tabella con lista di utenti con accesso al dispositivo
Ogni utente ha i permessi che gli sono concessi sul sys

Capability = token per rappresentare un oggetto sul quale si ha accesso

UNIX: x file aperti/socket/...

- cookie web
- file aperti
- SYSCALL OPEN ⟶ identificativo univoco che può essere riconosciuto dal sistema x identificare i file

- possono essere crittografate/segmentate

Sistema Mandatory Access Control

x regolare l'accesso di utenti diversi allo stesso file
x gestire i permessi in base a regole e livelli di sicurezza prestabiliti

modello Bell-LaPadula

- ogni Soggetto e ogni oggetto hanno un livello di sicurezza
- un Soggetto può leggere oggetti a livelli di sicurezza ≤ al suo
- un Soggetto può scrivere oggetti a livelli di sicurezza ≥ al suo

modello Biba

- ogni Soggetto e ogni oggetto hanno un livello di sicurezza
- un Soggetto può scrivere oggetti a livelli di sicurezza ≤ al suo
- un Soggetto può leggere oggetti a livelli di sicurezza ≥ al suo

File System

- gestiscono il sistema per lo scambio di dati locali
- analizza il proprietario del file, il gruppo cui appartiene ed altre proprietà eventuali

SUPER UTENTE

è un utente che può eseguire qualsiasi operazione dalla radice del sistema
ROOT su Linux
SYSTEM su Windows

▷ POSSONO CAMBIARE I PERMESSI AD ALTRI UTENTI/FILE con chmod

3PERMESSI SPECIALI

- setuid
- setgid
- Sticky BIT → concede di eliminare file all'interno della cartella marcata solo dal suo proprietario
  ↳ a prescindere dei permessi assegnati in precedenza

Buffer Overflow — vulnerabilità di sicurezza

> avviene quando un programma scrive dati in eccesso ai limiti del buffer andando a sovrascrivere un'area di memoria adiacente allo STACK

* consente di eseguire codice dannoso
* prendere il controllo del sistema senza la nostra volontà

get (...)
caratteri che alterano la visualizzazione 0x0A

Stack canaries — controllo per l'integrità dei dati posizionato nello STACK

Authenticated pointers — crittazione AES dei puntatori

Address Source Layout Randomization — randomizzare i vari indirizzi → rendere non riconoscibili i veri indirizzi

Memory Corruption

* ARBITRARY READ — espone parte della memoria mappata al processo → Trudy può conoscere la presenza di CANARINI o di puntatori ASLR riuscendo a bypassarle
* ARBITRARY WRITE — espone parte della memoria mappata al processo → Trudy può SCRIVERE su OGNI PARTE della MEMORIA MAPPATA ← * GOT attack sul file di caricamento delle dynamic lib.
* ARBITRARY EXECUTION — espone parte della memoria mappata al processo → Trudy può ESEGUIRE ogni parte della mem mappata

Librerie dinamiche

= parti di codice — inserito nel binario PRIMA del run time → CARICATE in locazioni di memoria CASUALI → usando un .got → APRE una vulnerabilità di MEM CORRUPTION → Posso cambiare la locazione di memoria DOVE il programma crede che ci sia una funzione

✓ condivisione del codice da Librerie Solo quando necessario
✓ ridurre le dimensioni del binario
✓ aggiornare la libreria senza dover ricompilare tutti i programmi che la EXEC

Printf (...) — VULNERABILITÀ → utente ha alta manovrabilità sul contenuto da passare → gets (...), %s, %n

* FORTIFY → aggiunge TEST al compiler → RIMUOVE Buffer Overflow
* Position Indipendent Executable — RAND (codice di funzioni vulnerabili) => programma inizia da un indirizzo CASUALE
* Partial RELRO — applicata a .got .plt
  Relocation Read-Only → cambia il layout di memoria
  × RINFORZARE
  × rendere il file READONLY => global var ESPORTATE sono protette

Laboratorio

Hacker

persona con competenze informatiche che le usa per superare un ostacolo o penetrare all'interno di un sistema tramite vie alternative

- **WHITE HAT** — sono i "buoni" cercano falle di sicurezza nei sistemi, con approvazione e consenso dei proprietari del sistema.
- **BLACK HAT** — sono i "cattivi", cercano falle di sicurezza nei sistemi per tornaconto personale

Hash

funzioni crittografiche che creano un codice univoco da un INPUT, di lunghezza diversa.

SCOPO: integrità e confidenzialità

- veloci da computare
- OUTPUT di piccole dimensioni, sempre differente ∀ INPUT
- 2 INPUT ≠ ⟼ 2 OUTPUT ≠

! IRREVERSIBILE

* **Brute Force** — provare tutte le combinazioni possibili, fino a quando non si trova quella giusta
* **Precomputed** — ma precomputo le hash più comuni ⟶ x comparare
* **Dictionary** — provare tutte le parole di una lista

Salting

aggiunta di dati casuali al PT x mascherare il vero significato

=> PRECOMPUTED non può avvenire

ECB mode

blocchi PT identici = blocchi CT identici ! PERICOLOSO

manca di DIFFUSIONE

CBC mode

attribuisce ad ogni blocco CT una key ≠ ⇒ $a = b \rightarrow C(a) \neq C(b)$