Modulo 1: Crittografia Simmetrica, Sicurezza Perfetta e Definizioni Formali

Sommario

Questa dispensa analizza i fondamenti della crittografia simmetrica, partendo dai concetti base fino alle definizioni rigorose di sicurezza necessarie per comprendere i moderni standard crittografici.

Indice

1 Fondamenti di Crittografia Simmetrica

La crittografia simmetrica rappresenta la forma più elementare di cifratura. La sua caratteristica distintiva è la **simmetria della chiave**: la stessa identica chiave K viene utilizzata sia per cifrare che per decifrare. Questo si contrappone alla crittografia asimmetrica (o a chiave pubblica), dove le chiavi sono distinte.

Il goal della **crittografia** è: proteggere la confidenzialità dei dati ***TRASFORMANDOLI*** in qualcosa di incomprensibile. La trasformazione però deve essere in qualche modo *reversibile*.

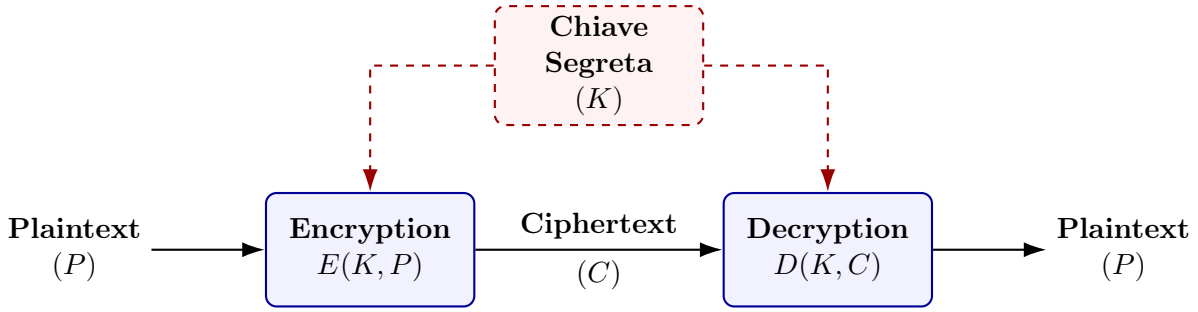

Figura 1: Modello funzionale della Crittografia Simmetrica: la stessa chiave K viene usata per E e D.

1.1 Definizioni e Modello Funzionale

Possiamo formalizzare un cifrario (*cipher*) come un insieme di due funzioni:

1. **Encryption (E):** Trasforma il messaggio in chiaro (Plaintext, P) in un messaggio cifrato (Ciphertext, C) utilizzando una chiave segreta K.

$$C = E(K, P)$$

2. **Decryption (D):** Ricostruisce il plaintext originale partendo dal ciphertext e dalla stessa chiave K.

$$P = D(K, C)$$

Nel linguaggio comune, spesso usiamo il termine "cifrario" per riferirci specificamente alla funzione di cifratura. Un cifrario può essere visualizzato come una "scatola" che prende in input P e K e produce C in output.

1.2 Cifrari Classici: La Sicurezza Debole

I primi tentativi storici, come i **Cifrari a Sostituzione**, operavano sostituendo ogni lettera del plaintext con un'altra secondo una regola fissa.

- **Vulnerabilità:** Sebbene lo spazio delle chiavi sembri ampio (26! combinazioni), questi cifrari preservano la struttura statistica del linguaggio.
- **Criptanalisi:** Un attaccante (spesso definito "analfabeta" nel contesto della sicurezza moderna per indicare che non necessita di matematica avanzata) può rompere il codice usando l'**analisi delle frequenze**. Lettere frequenti nel plaintext (es. 'E' in inglese o 'A'/'E' in italiano) appariranno con la stessa frequenza nel ciphertext, permettendo la decifrazione senza conoscere la chiave.

2 La Sicurezza Perfetta (One-Time Pad)

Per ottenere una sicurezza inviolabile, dobbiamo eliminare qualsiasi legame statistico tra plaintext e ciphertext. Questo ci porta al **One-Time Pad (OTP)**, l'unico cifrario che garantisce la **Perfect Secrecy**.

2.1 Funzionamento: L'Operazione XOR

L'OTP non usa complessi algoritmi di mescolamento, ma si basa sull'operazione logica **XOR** ($\oplus$), definita come:

- $0 \oplus 0 = 0$
- $0 \oplus 1 = 1$
- $1 \oplus 0 = 1$
- $1 \oplus 1 = 0$

Il processo è il seguente:

- **Cifratura:** Si fa lo XOR bit a bit tra il plaintext P e una chiave casuale K della stessa lunghezza.
$$C = P \oplus K$$
- **Decifratura:** Si fa nuovamente lo XOR tra il ciphertext C e la chiave K.
$$P = C \oplus K$$

Esempio Numerico

Se abbiamo il messaggio $P = 01101101$ e la chiave $K = 10110100$:

$$C = 01101101 \oplus 10110100 = 11011001$$

Per decifrare:

$$P = 11011001 \oplus 10110100 = 01101101$$

2.2 Perché è Sicuro? (L'Intuizione di Shannon)

Claude Shannon dimostrò negli anni '40 che l'OTP è **incondizionatamente sicuro**. Anche un attaccante con potenza di calcolo infinita non può romperlo.

Il motivo: Se la chiave K è perfettamente casuale, il ciphertext risultante C è indistinguibile da una stringa casuale.

Immaginiamo un attaccante che intercetta C. Poiché ogni bit della chiave ha il 50% di probabilità di essere 0 o 1, ogni bit del plaintext ha uguale probabilità di essere stato trasformato. Sapere C non dà alcuna informazione su P (eccetto la lunghezza). L'attaccante non impara nulla che non sapesse già prima di vedere il ciphertext.

2.3 I Limiti Pratici dell'OTP

Nonostante la perfezione teorica, l'OTP è "totalmente scomodo" per l'uso quotidiano:

1. **Lunghezza della Chiave:** La chiave deve essere lunga *quanto* il messaggio. Per cifrare un hard disk da 1 TB, serve una chiave da 1 TB.
2. **Gestione della Chiave:** Generare e distribuire in sicurezza chiavi così grandi è un problema logistico enorme.
3. **Malleabilità (Attacco all'Integrità):** L'OTP garantisce confidenzialità ma NON integrità.
    - Se l'attaccante intercetta $C = P \oplus K$ e vuole modificare il plaintext, può calcolare $C' = C \oplus \Delta$.
    - Il destinatario decifrerà $P' = P \oplus \Delta$.

- *Esempio:* Se P è "Bob" e l'attaccante vuole che diventi "Eve", può modificare il ciphertext senza conoscere la chiave, e il destinatario non se ne accorgerà.

2.4 Il Pericolo Mortale del Riutilizzo (Key Reuse)

Il nome "One-Time" non è un suggerimento, è un obbligo. Se si usa la stessa chiave K per due messaggi P_1 e P_2, la sicurezza crolla istantaneamente:

$$C_1 = P_1 \oplus K$$
$$C_2 = P_2 \oplus K$$

L'attaccante calcola lo XOR dei due ciphertext:

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2$$

La chiave si cancella! L'attaccante ottiene lo XOR dei due messaggi in chiaro, che è facilissimo da separare usando tecniche statistiche o conoscendo parti di uno dei messaggi. Questo viene chiamato *Known Plaintext Attack (KPA).*

3 Definizioni di Sicurezza (Security Notions)

Poiché l'OTP è impraticabile, usiamo cifrari computazionalmente sicuri. Ma cosa significa "sicuro"? Non possiamo usare definizioni vaghe. Dobbiamo combinare un **Obiettivo di Sicurezza** con un **Modello di Attacco**.

3.1 Probabilità in Crittografia

La sicurezza si misura in probabilità. Con una chiave di n bit, ci sono 2^n chiavi possibili. La probabilità di indovinare la chiave a caso è $1/2^n$.

Un cifrario è sicuro se la probabilità di successo di un attacco non è significativamente maggiore di $1/2^n$ (o $1/2$ nel caso di distinguere tra due messaggi).

3.2 Modelli di Attacco (Attack Models)

Definiscono le capacità dell'attaccante (Adversary). Sono ordinati dal più debole al più forte:

1. **Ciphertext-Only Attack (COA):** L'attaccante vede solo il ciphertext. È un osservatore passivo.
2. **Known-Plaintext Attack (KPA):** L'attaccante conosce alcune coppie (plaintext, ciphertext). Utile se l'attaccante sa che ogni email inizia con "Gentile Cliente".
3. **Chosen-Plaintext Attack (CPA):** L'attaccante è **attivo**. Può scegliere dei plaintext a suo piacimento e ottenere dal sistema i corrispondenti ciphertext. Questo simula scenari reali (es. iniettare dati in un sistema web per vedere come vengono cifrati).
4. **Chosen-Ciphertext Attack (CCA):** L'attaccante può scegliere dei ciphertext e ottenere i corrispondenti plaintext (accesso a un oracolo di decifratura). Modello molto forte, necessario per garantire l'integrità.

3.3 Obiettivi di Sicurezza (Security Goals)

Cosa vogliamo impedire all'attaccante di fare?

1. **Indistinguibilità (IND):** Il ciphertext deve sembrare rumore casuale. L'attaccante non deve poter distinguere quale di due messaggi è stato cifrato (vedi IND-CPA sotto).
2. **Non-Malleabilità (NM):** L'attaccante non deve poter modificare il ciphertext per creare un nuovo plaintext correlato in modo significativo all'originale (come visto nel difetto dell'OTP).

4 Sicurezza Semantica (IND-CPA)

La definizione moderna "standard" de facto per la confidenzialità è **IND-CPA** (**IND**istinguishability under **C**hosen **P**laintext **A**ttack).

4.1 Il Gioco IND-CPA

La sicurezza viene verificata tramite un esperimento mentale ("gioco") tra un Challenger e un Avversario:

1. **Fase di Apprendimento:** L'Avversario può chiedere la cifratura di qualsiasi messaggio (potere CPA).
2. **Sfida (Challenge):** L'Avversario sceglie due messaggi di uguale lunghezza, M_0 e M_1.
3. **Cifratura:** Il Challenger lancia una moneta (bit $b \in \{0, 1\}$), cifra M_b e invia il ciphertext C^* all'Avversario.
4. **Ipotesi:** L'Avversario deve indovinare se C^* contiene M_0 o M_1.

4.2 Condizione di Vittoria

Il cifrario è sicuro se l'Avversario non può indovinare meglio del caso puro.

$$P(\text{Indovinare}) \leq \frac{1}{2} + \epsilon$$

Dove ϵ è trascurabile. Se l'attaccante vince anche solo con probabilità 0.51, il sistema è rotto.

4.3 Cifratura Randomizzata vs Deterministica

Il gioco IND-CPA implica una verità fondamentale: la cifratura deterministica è insicura.

Se cifrare "CIAO" produce sempre lo stesso codice esadecimale, l'Avversario vince sempre il gioco:

1. Riceve la sfida C^*.
2. Usa il suo potere CPA per cifrare M_0 e ottenere C_{test}.
3. Se $C^* == C_{test}$, sa che il messaggio era M_0.

Per essere IND-CPA sicuro, l'algoritmo deve essere Randomizzato:

$$C = E(K, R, P)$$

Dove R è un valore casuale nuovo per ogni cifratura (spesso chiamato IV o Nonce).

In questo modo, cifrare M_0 due volte produce due ciphertext diversi (C_a e C_b). L'Avversario non può più confrontare i ciphertext per vincere il gioco.

5 Generazione di Chiavi (TRNG vs PRNG)

Tutta la sicurezza discussa sopra dipende dalla qualità della casualità (K o R).

- **TRNG (True Random Number Generators):** Estraggono casualità da fenomeni fisici (rumore termico, decadimento radioattivo). Essenziali per generare le chiavi, ma lenti.
- **PRNG (Pseudo-Random Number Generators):** Algoritmi deterministici che espandono un seme iniziale. Veloci, ma se il seme è debole o lo stato interno viene scoperto, l'attaccante può prevedere tutti i numeri futuri.

Case Study (Attacco RFID): Le slide mostrano un esempio dove un PRNG debole ha permesso a un attaccante di calcolare le chiavi future osservando le differenze (XOR) tra messaggi consecutivi, distruggendo la sicurezza del protocollo.