# Elementi di Crittografia

Appunti del Corso (A.A. 2025-2026)

24 ottobre 2025

## 1 Introduzione e Definizioni Fondamentali

Il corso di **Elementi di Crittografia** è un insegnamento della Laurea Magistrale e richiede come prerequisiti una solida base di *Algoritmi*, *Matematica Discreta* e *Teoria della Probabilità* acquisita durante la Triennale. Non è richiesta una conoscenza pregressa di crittografia.

### 1.1 Etimologia e Distinzioni

Il termine **Crittografia** deriva da due parole greche:

- **Kryptos** ($\kappa\rho\upsilon\pi\tau o\acute{\varsigma}$): Nascosto.
- **Graphia** ($\gamma\rho\alpha\phi\acute{\imath}\alpha$): Scrittura.

Storicamente è stato utilizzato anche il termine **Crittologia**, dove la seconda parte deriva da **Logos** ($\lambda\, o\acute{\gamma} o\varsigma$), che significa "discorso" o "studio". La distinzione tra i due termini è fondamentale per definire l'ambito di studio.

| Termine | Significato |
|---|---|
| **Crittografia** | La parte **costruttiva**: Progettazione e realizzazione di schemi crittografici (Scrittura nascosta). |
| **Criptoanalisi** | La parte **distruttiva**: Sforzi volti a sovvertire o "rompere" le proprietà di uno schema crittografico (Risolvere codici). |
| **Crittologia** | L'unione degli aspetti costruttivi e distruttivi: Crittografia + Criptoanalisi (Studio del nascosto). |

Tabella 1: Distinzioni fondamentali tra i campi di studio.

### 1.2 Definizioni Storiche vs. Definizione Moderna

Le definizioni storiche, pur essendo corrette per la millenaria origine della disciplina, non catturano l'ampiezza attuale del campo, sviluppatasi in particolare a partire dagli anni '70.

- **Vocabolario Treccani (Storica):** *"Crittografia: l'insieme delle teorie e delle tecniche (manuali, meccaniche o elettroniche) che permettono di cifrare un testo in chiaro..."*
- **Oxford Dictionary (Storica):** *"Cryptography: the art of writing or solving codes."*

Queste definizioni pongono l'accento sulla sola **Confidenzialità** (il "cifrare" un testo per renderlo inintelligibile).

## 2 La Transizione: Crittografia da Arte a Scienza

La crittografia moderna è lo strumento che permette di replicare nel mondo digitale (fatto di **sequenze di bit**) i meccanismi di sicurezza e fiducia che usiamo nella vita reale.

### 2.1 Il Concetto di Fiducia nel Mondo Reale

Nella vita di tutti i giorni, la sicurezza di operazioni delicate è garantita da **dispositivi fisici** o **parti fidate** (terze parti).

| **Attività Reale** | **Meccanismo di Protezione** |
|---|---|
| Conversare privatamente | Ambienti riservati / Protetti |
| Avere un diario segreto | Lucchetti / Cassaforte |
| Prelevare soldi dal conto | Bancomat / PIN |
| Votare alle elezioni | Cabine elettorali (Privacy) / Schede speciali (Integrità) |
| Compravendite | Notai / Ufficiali pubblici riconosciuti (Correttezza) |

Il presupposto di tutte queste forme di protezione è la **Fiducia** che riponiamo in tali dispositivi o enti. Poiché la società si è evoluta verso un mondo interconnesso e digitale, l'obiettivo è riprodurre queste garanzie di fiducia (sicurezza e privacy) nel nuovo contesto.

### 2.2 Definizione di Crittografia Moderna

La crittografia moderna è lo strumento che permette di ricreare digitalmente i meccanismi di fiducia **senza** ricorrere a dispositivi fisici o riducendo al minimo l'uso di parti fidate (es. blockchain, smart contracts).

**Definizione (Crittografia Moderna)** La Crittografia Moderna è lo **studio delle tecniche matematiche** utili a proteggere:

1. L'**informazione digitale** (o qualsiasi cosa essa rappresenti).
2. I **sistemi di elaborazione**.
3. Le **computazioni distribuite**.

Il tutto **in presenza di un avversario**, ovvero di un ente il cui obiettivo è sovvertire le funzionalità o le proprietà dei protocolli.

### 2.2.1 Implicazioni della Crittografia Moderna

La crittografia **nasce in risposta** alla presenza di avversari: se vivessimo in un mondo ideale non servirebbe.

- **Costo Aggiuntivo:** Progettare applicazioni in modo da preservare la funzionalità in un **ambiente ostile** (con avversari) comporta un costo aggiuntivo rispetto alle applicazioni progettate per ambienti ideali, risultando spesso più **lente** e **costose**.
- **Ambienti Avversari:** Un protocollo preserva la sua funzionalità in un ambiente avversario se, nonostante i tentativi di attacco (es. falsificazione del voto, scoperta dell'identità), gli obiettivi di sicurezza vengono mantenuti.

# 3 Contesto della Cifratura Simmetrica

Gli schemi di cifratura si dividono in due macro-categorie:

- **A chiave simmetrica (Symmetric-key setting):** La stessa chiave $k$ viene utilizzata sia per cifrare che per decifrare. Questo è il modello su cui ci concentreremo nella prima parte del corso.
- **A chiave asimmetrica (Asymmetric-key / Public-key setting):** Si utilizzano due chiavi diverse, una pubblica per cifrare e una privata per decifrare.

Il contesto a chiave simmetrica ha due applicazioni canoniche:

1. **Comunicazione sicura:** Due parti, separate "nello spazio", usano la chiave condivisa per proteggere i messaggi scambiati su un canale insicuro.
2. **Archiviazione sicura (Storage):** Una singola parte comunica con se stessa "nel tempo". Usa una chiave per cifrare dati da archiviare (es. su un hard disk) e la stessa chiave per decifrarli in un momento successivo.

# 4 Le Garanzie della Crittografia Moderna

La crittografia moderna va ben oltre la semplice confidenzialità (cifratura). Essa garantisce un insieme di proprietà fondamentali per i dati e i protocolli digitali.

## 4.1 Proprietà di Sicurezza Primarie

1. **Confidenzialità:** Garantisce che il contenuto dell'informazione non sia intelligibile a terze parti non autorizzate.
2. **Integrità dei Dati:** Garantisce che i dati (memorizzati o trasmessi) non siano stati modificati, in particolare a seguito di un **attacco avversario** (a differenza della correzione degli errori che gestisce errori accidentali).
3. **Autenticità:**
   - **Dei Dati:** Certifica che i dati siano stati prodotti dall'ente/persona che si presume sia l'origine (es. Firma Digitale).
   - **Delle Parti:** Assicura che, quando si dialoga con un'altra parte, questa sia effettivamente la persona (o il sistema) che si dichiara di essere.

## 4.2 Primitive e Protocolli Evoluti

La crittografia moderna rende possibili protocolli apparentemente impossibili da realizzare, gestendo problemi che non esistevano nel mondo analogico.

### 4.2.1 Scambio Sicuro di Chiavi

Poiché viviamo in un mondo in cui si interagisce con persone mai incontrate, i protocolli di **Scambio Sicuro di Chiavi (Key Exchange)** permettono a due parti di stabilire un'informazione segreta condivisa (la chiave crittografica) per poter poi comunicare in modo sicuro e confidenziale.

### 4.2.2 Condivisione di Segreti (Secret Sharing)

Permette di dividere un segreto $S$ in $n$ parti (share), tali che solo un sottoinsieme di almeno $k$ parti (soglia $k$) sia in grado di ricostruire il segreto, mentre un numero minore di $k$ parti non ottiene **alcuna informazione** su $S$.

### 4.2.3 Sistemi di Prova a Conoscenza Zero (Zero-Knowledge Proofs - ZKP)

Un **Sistema di Prova** generalizza il concetto classico di dimostrazione (Teorema + Prova) a un'interazione dinamica tra un **Provatore** ($P$) e un **Verificatore** ($V$).

Un ZKP è un protocollo che permette al Provatore di convincere il Verificatore che un certo enunciato ($P$ conosce $x$) è vero, **senza fornirgli alcuna conoscenza aggiuntiva** (la Prova) oltre alla veridicità dell'enunciato stesso.

$$\text{Output di } V : \begin{cases} \text{Vero} & \text{(L'enunciato è corretto)} \\ \text{Falso} & \text{(L'enunciato è scorretto)} \end{cases}$$

**Esempio: Autenticazione** Il Provatore vuole dimostrare di conoscere la propria password al Verificatore (server) senza che la password esca mai dal suo dispositivo. Il Verificatore riceve solo un bit: autenticazione riuscita o fallita.

### 4.2.4 Calcolo Sicuro di Funzioni (Secure Multi-Party Computation - MPC)

È il modello più generale sviluppato dalla crittografia moderna. Dati $n$ partecipanti $P_1, \ldots, P_n$ e una funzione $f$, dove ogni $P_i$ ha un input privato $x_i$. L'obiettivo è calcolare l'output $y = f(x_1, \ldots, x_n)$ in modo che:

- **Correttezza/Sicurezza:** La funzione $f$ sia calcolata correttamente.
- **Privacy:** Gli input individuali $x_i$ rimangano segreti.

**Esempio: Voto Elettronico**

- $P_i$: Elettore.
- $x_i$: Voto individuale ($\in \{0, 1\}$).
- $f$: Funzione di maggioranza $f(x_1, \ldots, x_n) = \text{Maggioranza}(\sum x_i)$.

Il protocollo MPC garantisce che il risultato (la maggioranza) sia corretto e che il voto di ciascun elettore $x_i$ rimanga segreto.

# 5 Transizione da Arte a Scienza: I Pilastri

Storicamente, gli schemi di cifratura erano progettati **ad hoc** (in modo artigianale) e valutati in base all'ingegnosità del progetto e alla **difficoltà percepita** di rottura, come la non facile applicabilità dell'analisi delle frequenze (riscoperta nel Medioevo, ma già nota nella cultura araba circa 1000 anni prima di Vigenère).

Nella crittografia classica, mancava:

1. Una **nozione condivisa** di cosa significasse per uno schema essere "sicuro".
2. Un modo per **produrre evidenza** (prove) della sicurezza.

La **Crittografia Moderna** sposta il confine dall'arte alla **scienza** attraverso tre elementi fondamentali:

1. **Definizioni Rigorose:** Specificano la **funzionalità** desiderata e la **nozione di sicurezza** in modo formale.
2. **Dimostrazioni (Prove):** Utilizzano strumenti matematici per dimostrare che una costruzione specifica soddisfa la definizione.
3. **Assunzioni Computazionali:** Sono congetture sulla **difficoltà di risolvere** certi problemi computazionali per i quali non si conoscono algoritmi risolutivi efficienti.

La maggior parte dei risultati di sicurezza sono **condizionati** alla veridicità di tali assunzioni.

**Risultati Condizionati:** Una dimostrazione crittografica tipica afferma: "Se l'Assunzione $X$ è vera (es. il problema è difficile), allora la Costruzione $\Pi$ raggiunge la Definizione $Z$." Questo è noto come **riduzione di sicurezza**.

### 5.1 Relazione tra Prove di Sicurezza e Mondo Reale

È fondamentale non sopravvalutare ciò che una prova di sicurezza garantisce. Le garanzie fornite sono sempre relative alla definizione di sicurezza adottata e alle assunzioni fatte. L'efficacia di una prova dipende in modo cruciale da quanto bene la definizione e il modello delle minacce riescono a catturare le complessità e le vulnerabilità del mondo reale in cui il sistema verrà impiegato. Una prova, quindi, non garantisce una sicurezza "assoluta", ma fornisce una forte evidenza che, *se il modello è corretto e le assunzioni sono valide*, allora nessun avversario che agisce entro i limiti del modello può avere successo.

## 6 Principio 1: Definizioni Rigorose e Segretezza

Le definizioni sono essenziali per la progettazione, la valutazione e la comparazione delle primitive. Una definizione rigorosa include:

1. **Garanzie di Sicurezza (*Security Goal*):** La proprietà che si vuole ottenere (es. Confidenzialità).
2. **Modello delle Minacce (*Threat Model*):** Il potere e le risorse di cui dispone l'avversario.

## 6.1 Definizione Corretta di Segretezza

Tentativi intuitivi di definire la sicurezza (es. "impossibile recuperare la chiave" o "recuperare ogni singolo carattere") falliscono perché non riescono a prevenire attacchi che rilasciano informazioni parziali, ma **significative** (es. se lo stipendio cifrato è maggiore di una soglia).

**Esempi di Definizioni Insufficienti.** Per capire la difficoltà nel formulare una buona definizione, consideriamo alcuni tentativi:

- *"Dovrebbe essere impossibile per un attaccante recuperare la chiave di cifratura?"* Questa definizione è debole. Lo schema $Enc_k(m) = m$ non fornisce alcuna informazione sulla chiave $k$, ma è palesemente insicuro poiché il messaggio è in chiaro.
- *"Dovrebbe essere impossibile recuperare l'intero testo in chiaro?"* Anche questa è insufficiente. Uno schema che protegge un database ma rivela il 90% del suo contenuto non può essere considerato sicuro [31].
- *"Dovrebbe essere impossibile recuperare qualsiasi carattere del messaggio in chiaro?"* Sembra una definizione migliore, ma non basta. In un database finanziario, uno schema potrebbe non rivelare i valori esatti ma far trapelare se una transazione è maggiore o minore di una certa soglia, un'informazione parziale ma critica.

Questi esempi mostrano che una definizione di sicurezza robusta deve escludere il rilascio di **qualsiasi informazione utile** sul messaggio in chiaro, al di là di ciò che l'avversario già sa.

**Obiettivo di Segretezza (Intuitivo):** Uno schema di cifratura sicuro deve **escludere il rilascio di qualsiasi informazione utile** da parte del cifrato. Il cifrato non dovrebbe dare **nessun vantaggio** all'avversario.

L'idea è che, qualunque sia l'informazione *a priori* dell'attaccante, l'osservazione del cifrato non deve alterare la sua conoscenza (probabilità stimata) sul contenuto del messaggio.

### 6.1.1 Formalizzazione di Shannon (Perfetta Segretezza)

La formalizzazione più perfetta di questa nozione, dovuta a Claude Shannon, richiede che la probabilità che un messaggio in chiaro $M$ sia uguale a un valore $m$, dato il cifrato $C = c$, sia identica alla probabilità *a priori* del messaggio, $P[M = m]$.

**Definizione** (Segretezza Perfetta)**.** *Uno schema di cifratura a chiave privata* $\Pi = (Gen, Enc, Dec)$ *è* ***perfettamente segreto*** *se, per ogni spazio dei messaggi* $M$ *e per qualsiasi distribuzione di probabilità sui messaggi, per tutti gli* $m \in M$ *e per tutti i cifrati* $c \in C$*, risulta:*

$$Pr[M = m \mid C = c] = Pr[M = m]$$

*In termini semplici: il cifrato c non fornisce alcuna informazione aggiuntiva sul messaggio m rispetto a quella posseduta a priori dall'avversario.*

# 7 Modello delle Minacce: Potere e Scenari d'Attacco

Il modello delle minacce definisce i limiti e le risorse dell'avversario senza specificare la sua **strategia d'attacco**, che è libera.

## 7.1 Potere Computazionale

Si considerano due modelli principali per il potere dell'avversario:

| Modello | Descrizione |
|---|---|
| **Avversario Illimitato (*Unbounded*)** | Possiede potere computazionale **infinito** (esegue computazioni esponenziali in tempo ragionevole). È un'assunzione fortissima necessaria per la **perfetta segretezza**. |
| **Avversario Limitato (*Bounded/Polinomiale*)** | Possiede un potere computazionale **ragionevole**, limitato agli algoritmi in **tempo polinomiale**. È il modello dominante nella **crittografia moderna** (sicurezza computazionale). |

Tabella 2: Modelli per il potere computazionale dell'avversario.

## 7.2 Scenari d'Attacco in Cifratura

Gli scenari descrivono le informazioni aggiuntive (coppie messaggio/cifrato) che l'avversario può raccogliere prima di tentare di decifrare il cifrato di suo interesse (sfida). In tutti gli scenari, si presuppone che i cifrati siano stati prodotti con la **stessa chiave segreta** $K$ usata nella comunicazione in corso.

| Scenario | Potere dell'Avversario |
|---|---|
| **Ciphertext Only (COA)** | Può solo **ascoltare** il canale e vedere il cifrato $C$. |
| **Known-Plaintext Attack (KPA)** | Acquisisce **coppie (Messaggio $M_i$, Cifrato $C_i$)** precedentemente prodotte con la chiave $K$. |
| **Chosen-Plaintext Attack (CPA)** | Può **scegliere i messaggi** $M_i$ e ottenere i cifrati $C_i$ corrispondenti. |
| **Chosen-Ciphertext Attack (CCA)** | Può **scegliere sia i messaggi** (per ottenere $C_i$) **sia i cifrati** $C'_j$ (per ottenere i messaggi $M'_j$ decifrati). Rappresenta l'attacco più forte. |

Tabella 3: Scenari d'attacco in un contesto di cifratura simmetrica.

**Nota Importante:** La sicurezza minima richiesta per uno schema di cifratura (quello usato negli standard) è che raggiunga una nozione di sicurezza rispetto ad attacchi di tipo **Chosen-Plaintext (CPA)**. Gli schemi che resistono anche agli attacchi **Chosen-Ciphertext (CCA)** sono considerati i più robusti e sono più difficili da progettare.

# 8 Richiami di Probabilità Fondamentali

## Spazi di Probabilità (Probability Spaces)

- **Spazio Campionario ($\Omega$):** L'insieme di tutti i possibili risultati di un esperimento casuale (eventi elementari).
- **Evento ($E$):** Un sottoinsieme di $\Omega$ ($E \subseteq \Omega$).
- **Funzione di Probabilità (Pr):** Una funzione che assegna ad ogni evento $E$ un valore tra 0 e 1, tale che:
  1. $\Pr[E] \geq 0$ per ogni evento $E$.
  2. $\Pr[\Omega] = 1$ (la probabilità che accada un risultato in $\Omega$ è 1).
  3. Se $E_1$ ed $E_2$ sono eventi mutualmente esclusivi ($E_1 \cap E_2 = \varnothing$), allora $\Pr[E_1 \cup E_2] = \Pr[E_1] + \Pr[E_2]$.

## Variabili Aleatorie (Random Variables)

**Definizione.** *Una **Variabile Aleatoria** ($X$) è una funzione $X : \Omega \to \mathbb{R}$ che assegna un valore numerico a ogni risultato dell'esperimento casuale. Nel contesto crittografico, spesso $X$ è una stringa di bit.*

- **Valore Atteso (Expected Value):** Sia $X$ una variabile aleatoria su uno spazio di probabilità finito. Il suo valore atteso è dato da:

$$E[X] = \sum_{x \in X(\Omega)} x \cdot \Pr[X = x]$$

## Ensemble di Distribuzioni e Parametro di Sicurezza

- **Parametro di Sicurezza ($n$):** Un intero $n$ passato come input $1^n$ agli algoritmi, che quantifica il livello di sicurezza. La sicurezza viene valutata al crescere di $n$.
- **Ensemble di Distribuzioni:** Una famiglia di distribuzioni di probabilità $\mathcal{D} = \{D_n\}_{n \in \mathbb{N}}$, dove $D_n$ è una distribuzione su un insieme $S_n$ associato al parametro di sicurezza $n$. (Esempio: $U_n$ è la distribuzione uniforme sulle stringhe di $n$ bit).

## Disuguaglianza di Markov

**Teorema** (Disuguaglianza di Markov)**.** *Per una variabile casuale $X$ non negativa e qualsiasi $\lambda > 0$, la probabilità che $X$ sia maggiore o uguale a $\lambda$ è limitata dal rapporto tra il suo valore atteso e $\lambda$:*

$$Pr[X \geq \lambda] \leq \frac{E[X]}{\lambda}$$

# 9 Crittografia Classica e Principi Fondamentali

Prima di addentrarci nelle definizioni rigorose della crittografia moderna, è utile analizzare alcuni schemi storici. Questi esempi, pur essendo insicuri secondo gli standard odierni, permettono di introdurre concetti chiave e di comprendere perché un approccio *ad hoc* alla progettazione di cifrari è destinato a fallire.

## 9.1 Il Principio di Kerckhoffs

Un principio fondamentale, enunciato da Auguste Kerckhoffs alla fine del XIX secolo, guida tutta la crittografia moderna.

> "Il metodo di cifratura non è richiesto che sia segreto e dovrebbe essere in grado di cadere nelle mani del nemico senza creare problemi."

Questo implica che **la sicurezza di uno schema crittografico deve basarsi unicamente sulla segretezza della chiave, non sulla segretezza dell'algoritmo** [680]. Le ragioni sono pratiche e strategiche:

- È più facile proteggere e cambiare una chiave (una stringa di bit) che un intero algoritmo.
- Permette la standardizzazione e lo scrutinio pubblico degli algoritmi, aumentandone l'affidabilità.

La pratica di affidarsi alla segretezza dell'algoritmo è nota come *security by obscurity* ed è universalmente considerata debole.

# 10 Cifrari a Sostituzione Monoalfabetica

Questi cifrari operano sostituendo ogni carattere del messaggio in chiaro con un altro carattere, secondo una regola fissa.

## 10.1 Il Cifrario di Cesare e lo Shift Cipher

Il Cifrario di Cesare, descritto da Svetonio, è un caso speciale dello Shift Cipher. Ogni lettera del messaggio viene sostituita dalla lettera che si trova 3 posti in avanti nell'alfabeto (A→D, B→E, ...).

**Definizione Formale (Shift Cipher).** Si mappa l'alfabeto inglese sui numeri $\{0, 1, ..., 25\}$. La chiave $k$ è un intero in questo insieme.

- **Enc**: $c_i = (m_i + k) \pmod{26}$.
- **Dec**: $m_i = (c_i - k) \pmod{26}$.

La debolezza di questo cifrario risiede nel suo spazio delle chiavi estremamente piccolo: ci sono solo 26 possibili chiavi. Un avversario può semplicemente provarle tutte. Questo attacco è noto come **ricerca esaustiva** o **attacco di forza bruta**. Da qui deriva il primo principio di sicurezza:

> **Sufficient Key-Space Principle:** Qualsiasi schema di cifratura sicuro deve avere uno spazio delle chiavi sufficientemente grande da rendere impraticabile un attacco per ricerca esaustiva. (Oggi, almeno $2^{80}$ elementi).

## 10.2 Un Attacco più Efficiente allo Shift Cipher

Oltre alla forza bruta, è possibile rompere lo Shift Cipher in modo più efficiente usando la statistica della lingua. Sia $p_i$ la frequenza della i-esima lettera dell'alfabeto inglese (es. $p_4 \approx 0.127$ per la 'e') e $q_i$ la frequenza osservata della i-esima lettera nel testo cifrato. Se la chiave usata è $k$, ci aspettiamo che la distribuzione delle frequenze del cifrato sia una versione "spostata" di quella originale, cioè $q_{i+k} \approx p_i$ per ogni $i$. L'attacco procede così:

1. Calcola il vettore delle frequenze $\vec{q} = (q_0, ..., q_{25})$ dal testo cifrato.
2. Per ogni possibile chiave $j \in \{0, ..., 25\}$, calcola una misura di correlazione tra il vettore delle frequenze del testo in chiaro $\vec{p}$ e il vettore $\vec{q}$ "spostato" all'indietro di $j$ posizioni. Una misura comune è $I_j = \sum_{i=0}^{25} p_i \cdot q_{i+j}$.
3. Il valore di $j$ che massimizza $I_j$ è la chiave più probabile. Per la lingua inglese, ci aspettiamo che per la chiave corretta $k$, il valore $I_k$ sia vicino a 0.065, mentre per chiavi errate sia significativamente diverso.

Questo attacco è facilmente automatizzabile e non richiede interpretazione umana del testo decifrato.

## 10.3 Cifrario per Sostituzione Monoalfabetica Generale

Per risolvere il problema dello spazio delle chiavi, si può usare una permutazione arbitraria dell'alfabeto come chiave. La chiave è una mappatura fissa, ad esempio:

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

Lo spazio delle chiavi è $26! \approx 2^{88}$, rendendo un attacco di forza bruta impraticabile. Tuttavia, questo cifrario è estremamente debole a causa degli **attacchi basati sull'analisi delle frequenze** [699].

- **Vulnerabilità:** Poiché ogni lettera del messaggio in chiaro viene mappata sempre sullo stesso carattere cifrato (es. 'e' diventa sempre 'D'), la distribuzione statistica delle frequenze delle lettere della lingua originale (es. l'inglese, dove 'e' è la più comune) si preserva nel testo cifrato.
- **Attacco:** Un avversario calcola le frequenze dei caratteri nel testo cifrato, le confronta con le frequenze note della lingua inglese e deduce le sostituzioni. Con un testo sufficientemente lungo, la decifrazione è semplice.

# 11 Cifrari a Sostituzione Polialfabetica: Vigenère

Per contrastare l'analisi delle frequenze, sono stati introdotti i cifrari polialfabetici, dove la stessa lettera del messaggio in chiaro può essere cifrata in modi diversi a seconda della sua posizione.

**Definizione (Cifrario di Vigenère).** La chiave $k$ è una parola di lunghezza $t$ (il *periodo*). Per cifrare un messaggio, la chiave viene ripetuta fino a coprire la lunghezza del messaggio. Ogni lettera del messaggio $m_i$ viene cifrata usando la lettera corrispondente della chiave estesa $k_j$ tramite uno Shift Cipher.

```
Messaggio: tellhimaboutme
Chiave:    cafecafecafeca
Cifrato:   VFPJIRENGZMLM
```

In questo modo, la 'e' di "tell" viene cifrata con la chiave 'f', mentre la 'e' di "me" viene cifrata con la chiave 'a', producendo caratteri cifrati diversi. Questo "appiattisce" la distribuzione delle frequenze, rendendo l'analisi monoalfabetica inefficace.

## 11.1 Attacchi al Cifrario di Vigenère

Ritenuto inattaccabile per secoli, il cifrario di Vigenère presenta una debolezza cruciale: la periodicità della chiave. L'attacco si svolge in due fasi:

1. **Determinare la lunghezza della chiave (t):**
   - **Metodo di Kasiski:** Si cercano sequenze di caratteri ripetute nel testo cifrato. La distanza tra queste ripetizioni è molto probabilmente un multiplo della lunghezza della chiave $t$. Calcolando il massimo comun divisore (MCD) di queste distanze, si ottiene un'ipotesi per $t$.

- **Indice di Coincidenza:** Un metodo più formale. Si divide il testo cifrato in $w$ colonne. Se $w$ è la vera lunghezza della chiave $t$, ogni colonna sarà un testo cifrato con un semplice Shift Cipher e la sua distribuzione di frequenze sarà simile a quella della lingua inglese (spostata). Altrimenti, se $w \neq t$, la distribuzione sarà quasi uniforme. Calcolando l'indice di coincidenza (una misura statistica) per diversi valori di $w$, il valore corretto $t$ emergerà chiaramente.

2. **Rompere i singoli Shift Cipher:** Una volta nota la lunghezza $t$, il testo cifrato viene diviso in $t$ sottotesti. Ognuno di questi è stato cifrato con una singola lettera della chiave (uno Shift Cipher) e può essere facilmente decifrato con l'analisi delle frequenze.

La debolezza dei cifrari classici ha reso evidente la necessità di un approccio più rigoroso e matematicamente fondato, che è alla base della crittografia moderna.

# 12 Introduzione alla Segretezza Perfetta

La Segretezza Perfetta rappresenta il **massimo livello di sicurezza** a cui uno schema di cifratura può aspirare. In questo ambito, ci si concentra sulla **confidenzialità** o **riservatezza** del messaggio, operando nel modello di **sicurezza a livello informazionale (*information-theoretic security*)**.

Il presupposto fondamentale è che l'avversario (Adv) disponga di **potere computazionale illimitato**. Ciò significa che l'avversario può eseguire qualsiasi strategia crittanalitica, anche quelle con un costo esponenziale in tempo, e la sicurezza non dipende dalla sua inefficienza, bensì da un'impossibilità matematica data dalla struttura dello schema stesso.

In questo contesto, la Crittografia Moderna si basa su tre pilastri: Definizioni Formali, Assunzioni (computazionali) e Prove di Sicurezza. La segretezza perfetta, essendo **incondizionata**, richiede soltanto Definizioni e Prove, **senza alcuna assunzione computazionale**.

## 12.1 Definizione Formale di Schema di Cifratura

Prima di definire la segretezza perfetta, formalizziamo la sintassi di uno schema di cifratura.

**Definizione** (Schema di Cifratura a chiave privata)**.** *Uno schema di cifratura a chiave privata è una tripla* $\Pi = (Gen, Enc, Dec)$ *di algoritmi* ***Probabilistici a Tempo Polinomiale (PPT)*** *tale che:*

1. $k \leftarrow Gen(1^n)$ *è un algoritmo probabilistico che prende in input il parametro di sicurezza* $1^n$ *e restituisce una chiave* $k \in K$*.*

2. *$c \leftarrow Enc_k(m)$ è un algoritmo probabilistico di cifratura che prende in input il messaggio $m \in \{0,1\}^*$ e la chiave $k \in K$, e restituisce un cifrato $c \in \{0,1\}^*$.*
3. *$m := Dec_k(c)$ è un algoritmo deterministico di decifratura che prende in input il cifrato $c \in \{0,1\}^*$ e la chiave $k \in K$, e restituisce il messaggio $m \in \{0,1\}^*$ (o un simbolo di errore $\perp$).*

***Correttezza:*** *Per ogni $n \in \mathbb{N}$, per ogni $k$ restituito da $Gen(1^n)$ e per ogni $m \in \{0,1\}^*$, risulta $Dec_k(Enc_k(m)) = m$, con probabilità 1.*

Di seguito viene enunciata una spiegazione informale della definizione per comprendere meglio i concetti:

- **Definizione:** Uno schema di cifratura è una tupla di tre algoritmi (Gen, Enc, Dec) e uno spazio dei messaggi $\mathcal{M}$.
  1. **Gen**: È un algoritmo probabilistico di generazione delle chiavi. Restituisce una chiave $k$ dallo spazio delle chiavi $\mathcal{K}$ secondo una certa distribuzione di probabilità.
  2. **Enc**: È un algoritmo (potenzialmente probabilistico) di cifratura. Prende in input una chiave $k \in \mathcal{K}$ e un messaggio $m \in \mathcal{M}$ e restituisce un cifrato $c$ nello spazio dei cifrati $\mathcal{C}$ .
  3. **Dec**: È un algoritmo deterministico di decifratura. Prende in input una chiave $k \in \mathcal{K}$ e un cifrato $c \in \mathcal{C}$ e restituisce un messaggio $m \in \mathcal{M}$.
- **Notazione:** Useremo le seguenti notazioni:
  - $k \leftarrow \text{Gen}()$: Assegnamento dell'output di un algoritmo probabilistico.
  - $c \leftarrow \text{Enc}_k(m)$: Cifratura probabilistica.
  - $m := \text{Dec}_k(c)$: Assegnamento dell'output di un algoritmo deterministico.
- **Condizione di Correttezza Perfetta:** Per ogni chiave $k \in \mathcal{K}$ e ogni messaggio $m \in \mathcal{M}$, deve valere $\text{Dec}_k(\text{Enc}_k(m)) = m$ con probabilità 1.

## 12.2 Variabili Aleatorie e Probabilità

Per analizzare la sicurezza, trattiamo chiavi, messaggi e cifrati come variabili aleatorie.

- **K**: La variabile aleatoria che denota la chiave scelta da Gen(). Assumeremo che la distribuzione sia uniforme su $\mathcal{K}$, salvo diversa indicazione.
- **M**: La variabile aleatoria che denota il messaggio in chiaro. La sua distribuzione, $Pr[M = m]$, riflette la conoscenza a priori dell'avversario (es. la frequenza con cui certi messaggi vengono inviati) e non dipende dallo schema di cifratura. Si assume che K e M siano indipendenti.
- **C**: La variabile aleatoria che denota il cifrato. La sua distribuzione è determinata dalle distribuzioni di K e M e dall'algoritmo Enc.

**Esempio di Calcolo Probabilistico con lo Shift Cipher.** Consideriamo lo Shift Cipher, dove $\mathcal{K} = \{0, \ldots, 25\}$ e Gen() sceglie $k$ uniformemente ($Pr[K = k] = 1/26$) [324]. Supponiamo che lo spazio dei messaggi sia $\mathcal{M} = \{a, z\}$ con la seguente distribuzione a priori: $Pr[M = a] = 0.7$ e $Pr[M = z] = 0.3$.

- **Qual è la probabilità che il cifrato sia 'B'?** Un cifrato 'B' (corrispondente al valore 1) si può ottenere in due modi:

  1. Cifrando $m = a$ (valore 0) con la chiave $k = 1$. La probabilità è $Pr[M = a \wedge K = 1] = Pr[M = a] \cdot Pr[K = 1] = 0.7 \cdot \frac{1}{26}$.
  2. Cifrando $m = z$ (valore 25) con la chiave $k = 2$. La probabilità è $Pr[M = z \wedge K = 2] = Pr[M = z] \cdot Pr[K = 2] = 0.3 \cdot \frac{1}{26}$.

  La probabilità totale è la somma: $Pr[C = B] = (0.7 \cdot \frac{1}{26}) + (0.3 \cdot \frac{1}{26}) = 1 \cdot \frac{1}{26} = \frac{1}{26}$.

- **Qual è la probabilità che il messaggio fosse 'a', dato il cifrato 'B'?** Usiamo il Teorema di Bayes :

$$Pr[M = a \mid C = B] = \frac{Pr[C = B \mid M = a] \cdot Pr[M = a]}{Pr[C = B]}$$

  Sappiamo che $Pr[C = B \mid M = a]$ è la probabilità di ottenere 'B' se il messaggio è 'a'. Questo accade solo se la chiave è $k = 1$, quindi $Pr[C = B \mid M = a] = Pr[K = 1] = 1/26$ [331]. Sostituendo i valori:

$$Pr[M = a \mid C = B] = \frac{(1/26) \cdot 0.7}{1/26} = 0.7$$

  Notiamo che $Pr[M = a \mid C = B] = Pr[M = a]$. In questo caso specifico, osservare il cifrato 'B' non ha cambiato la nostra conoscenza sul messaggio. Se questo vale per *ogni* messaggio e *ogni* cifrato, lo schema è perfettamente segreto.

## 12.3 Il Ruolo della Casualità (Randomness)

Non c'è segretezza senza casualità. La generazione di bit casuali è fondamentale perché la chiave segreta condivisa deve essere scelta a caso, altrimenti un avversario potrebbe prevederla.

**Generazione di Bit Casuali.** Le tecniche moderne procedono in due fasi:

1. **Raccolta di dati ad alta entropia:** Si raccolgono dati da sorgenti fisiche imprevedibili, anche se non uniformemente distribuiti. Esempi includono: ritardi di rete, tempi di accesso al disco, movimenti del mouse o rumore termico in componenti elettronici.
2. **Estrazione:** I dati raccolti vengono elaborati da un **estrattore di casualità** per produrre una sequenza di bit quasi indipendenti e uniformemente distribuiti.

**Esempio: Estrattore di Von Neumann.** Supponiamo di avere una moneta truccata, che produce 'Testa' (1) con probabilità $p$ e 'Croce' (0) con probabilità $(1-p)$. Per estrarre bit uniformi:

- Si lancia la moneta a coppie.
- Se la coppia è '01', si produce un bit '0'.
- Se la coppia è '10', si produce un bit '1'.
- Se la coppia è '00' o '11', non si produce alcun bit e si passa alla coppia successiva.

Questo metodo funziona perché la probabilità di ottenere '01' (che produce '0') è $(1-p)p$, e la probabilità di ottenere '10' (che produce '1') è $p(1-p)$. Le due probabilità sono identiche, garantendo che i bit prodotti siano uniformi.

# 13 Definizione Formale di Segretezza Perfetta

Sia $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ uno schema di cifratura con uno spazio dei messaggi $\mathcal{M}$. Sia $M$ la variabile casuale che descrive la distribuzione di probabilità dei messaggi in $\mathcal{M}$, e $C$ la variabile casuale che descrive la distribuzione dei cifrati in $\mathcal{C}$.

**Definizione** (Segretezza Perfetta - Definition 2.3 in Katz/Lindell)**.** *Un schema di cifratura* $\Pi$ *è* ***perfettamente segreto*** *se, per ogni distribuzione di probabilità per* $M$*, per ogni messaggio* $m \in \mathcal{M}$*, e per ogni cifrato* $c \in \mathcal{C}$ *tale che* $\Pr[C = c] > 0$*, si ha:*

$$\Pr[M = m \mid C = c] = \Pr[M = m] \tag{1}$$

Questa definizione implica che la **probabilità a posteriori** (dopo aver osservato il cifrato $c$) che sia stato inviato il messaggio $m$ è esattamente uguale alla **probabilità a priori** (prima di osservare il cifrato). In altre parole, **l'osservazione del cifrato non aggiunge alcuna informazione** alla conoscenza dell'avversario sul messaggio in chiaro.

**Definizione Equivalente.** Una definizione equivalente, spesso più semplice da usare nelle dimostrazioni, si basa sulla distribuzione dei cifrati.

> Uno schema di cifratura è perfettamente segreto se, per ogni coppia di messaggi $m_0, m_1 \in \mathcal{M}$ e per ogni cifrato $c \in \mathcal{C}$, vale:
>
> $$Pr[\text{Enc}_K(m_0) = c] = Pr[\text{Enc}_K(m_1) = c]$$
>
> dove la probabilità è calcolata sulla scelta casuale della chiave $K$.

Questo significa che la distribuzione di probabilità sui cifrati è identica indipendentemente da quale messaggio sia stato cifrato.

**Esperimento di Indistinguibilità ($PrivK^{eav}_{A,\Pi}$).** La nozione di indistinguibilità è formalizzata attraverso un "gioco" tra un avversario A e un challenger.

1. **Fase 1:** L'avversario A sceglie due messaggi distinti $m_0$, $m_1$ dallo spazio dei messaggi e li invia al challenger.
2. **Fase 2:** Il challenger genera una chiave casuale $k \leftarrow \text{Gen}()$, sceglie un bit $b \in \{0,1\}$ uniformemente a caso, e calcola il "cifrato di sfida" $c \leftarrow \text{Enc}_k(m_b)$.
3. **Fase 3:** Il challenger invia $c$ ad A.
4. **Fase 4:** A analizza $c$ e restituisce un bit $b'$ come sua ipotesi su quale messaggio sia stato cifrato.
5. **Successo:** L'esperimento ha successo (output 1) se $b' = b$. Altrimenti fallisce (output 0).

## 13.1 Segretezza Perfetta vs. Perfetta Indistinguibilità

Un concetto equivalente è quello di **Perfetta Indistinguibilità**.

**Definizione** (Perfetta Indistinguibilità - Definizione 2.6)**.** *Uno schema di cifratura* $\Pi$ *è* ***perfettamente indistinguibile*** *se per ogni avversario A, la probabilità che A abbia successo nell'esperimento di Indistinguibilità* ($PrivK^{eav}_{A,\Pi}$) *è pari a* $1/2$.

**Esempio: il Cifrario di Vigenère non è Perfettamente Indistinguibile.** Mostriamo con un esempio che il cifrario di Vigenère, per certi parametri, fallisce il test di indistinguibilità. Consideriamo uno schema di Vigenère per messaggi di due caratteri, dove la chiave (periodo) è una stringa di lunghezza 1 o 2, scelta con uguale probabilità (1/2).

- **Setup dell'Avversario A:**
  1. L'avversario A sceglie i due messaggi $m_0$ = "aa" e $m_1$ = "ab" e li invia al challenger.
  2. Il challenger sceglie un bit $b \in \{0,1\}$ a caso, genera una chiave $k$ e calcola il cifrato di sfida $c = \text{Enc}_k(m_b)$.
  3. A riceve $c = c_1 c_2$ e applica la seguente strategia: se $c_1 = c_2$, indovina $b' = 0$; altrimenti, indovina $b' = 1$.
- **Analisi del Successo:** La probabilità di successo di A è $Pr[\text{successo}] = \frac{1}{2}Pr[A \text{ vince} \mid b = 0] + \frac{1}{2}Pr[A \text{ vince} \mid b = 1]$.
  - **Caso b=0 (messaggio "aa"):** Il cifrato è $c = \text{Enc}_k(\text{"aa"})$.
    - Se la chiave ha lunghezza 1 (es. $k_1$), allora $c_1 = a + k_1$ e $c_2 = a + k_1$. Quindi $c_1 = c_2$. A indovina $b' = 0$ e vince. Questo accade con probabilità $1/2$.

* Se la chiave ha lunghezza 2 (es. $k_1k_2$), $c_1 = a + k_1$ e $c_2 = a + k_2$. $c_1 = c_2$ solo se $k_1 = k_2$, evento che ha probabilità 1/26. A vince se $k_1 = k_2$. Questo accade con probabilità $\frac{1}{2} \cdot \frac{1}{26}$.

$Pr[A \text{ vince} \mid b = 0] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{26} \approx 0.52$.

– **Caso b=1 (messaggio "ab"):** Il cifrato è $c = \text{Enc}_k(\text{"ab"})$. A vince se $c_1 \neq c_2$.

* Se la chiave ha lunghezza 1 (es. $k_1$), $c_1 = a + k_1$ e $c_2 = b + k_1$. Poiché $a \neq b$, allora $c_1 \neq c_2$ sempre. A indovina $b' = 1$ e vince. Questo accade con probabilità 1/2.
* Se la chiave ha lunghezza 2 (es. $k_1k_2$), $c_1 = a + k_1$ e $c_2 = b + k_2$. $c_1 \neq c_2$ a meno che non accada per caso che $a + k_1 = b + k_2$, cioè $k_2 - k_1 = a - b = -1 \pmod{26}$. Questo accade con probabilità 1/26. A vince se $k_2 - k_1 \neq -1 \pmod{26}$. Questo accade con probabilità $\frac{1}{2} \cdot (1 - \frac{1}{26})$.

$Pr[A \text{ vince} \mid b = 1] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 - \frac{1}{26}) = 1 - \frac{1}{52} \approx 0.98$.

La probabilità totale di successo per A è $\frac{1}{2}(0.52) + \frac{1}{2}(0.98) \approx 0.75$, che è significativamente maggiore di 1/2. Lo schema non è perfettamente indistinguibile.

**Lemma** (Equivalenza)**.** *Uno schema di cifratura* $\Pi$ *è perfettamente segreto* ***se e solo se*** *è perfettamente indistinguibile.*

Il fallimento della perfetta indistinguibilità, come mostrato per il cifrario di **Vigenere** per certi parametri, implica il fallimento della perfetta segretezza.

# 14 Il Cifrario One-Time Pad (OTP)

Il Cifrario OTP è l'esempio canonico di schema perfettamente segreto, storicamente noto come Vernam Cipher.

## 14.1 Definizione dello Schema

Si consideri uno spazio di messaggi $\mathcal{M} = \{0,1\}^\ell$, dove $\ell$ è la lunghezza del messaggio in bit.

1. **Generazione della Chiave (Gen):** Scegliere una chiave $K \in \mathcal{K} = \{0,1\}^\ell$ in modo **uniformemente casuale**. La chiave deve avere la stessa lunghezza del messaggio.
2. **Cifratura ($\text{Enc}_K$):** Dato il messaggio $m$ e la chiave $k$, il cifrato $c$ è calcolato come l'operazione bit a bit di XOR ($\oplus$):

$$c = \text{Enc}_k(m) = m \oplus k \tag{2}$$

3. **Decifratura ($\text{Dec}_K$):** Dato il cifrato $c$ e la chiave $k$, il messaggio $m$ è riottenuto grazie alla proprietà auto-inversa dell'operazione XOR ($x \oplus y \oplus y = x$):

$$m = \mathrm{Dec}_k(c) = c \oplus k = (m \oplus k) \oplus k = m \tag{3}$$

## 14.2 Dimostrazione della Segretezza Perfetta dell'OTP

**Teorema.** *Il cifrario* ***One-Time Pad*** *è perfettamente segreto.*

**Dimostrazione.** *Sia* $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^\ell$*, quindi* $| \mathcal{M} | = | \mathcal{C} | = | \mathcal{K} | = 2^\ell$*. Sia* $m \in \mathcal{M}$ *e* $c \in \mathcal{C}$ *un messaggio e un cifrato generico. Per la Segretezza Perfetta, dobbiamo dimostrare che* $\Pr[M = m \mid C = c] = \Pr[M = m]$*.*

*Per il* ***Teorema di Bayes****, abbiamo:*

$$\Pr[M = m \mid C = c] = \frac{\Pr[C = c \mid M = m] \cdot \Pr[M = m]}{\Pr[C = c]} \tag{4}$$

**Passaggio 1: Calcolo di** $\Pr[C = c \mid M = m]$

*Assumendo che il messaggio in chiaro sia* $m$*, il cifrato risultante è* $C = m \oplus K$*. Affinché il cifrato sia* $c$*, deve essere* $m \oplus K = c$*. Risolvendo per la chiave* $K$*, otteniamo l'unica chiave necessaria* $k^* = m \oplus c$*. Poiché la chiave* $K$ *è scelta in modo uniformemente casuale da* $\mathcal{K}$ *(con* $| \mathcal{K} | = 2^\ell$*), la probabilità di scegliere* $k^*$ *è:*

$$\Pr[C = c \mid M = m] = \Pr[K = m \oplus c] = \frac{1}{2^\ell} = \frac{1}{| \mathcal{K} |} \tag{5}$$

**Passaggio 2: Calcolo di** $\Pr[C = c]$

*La probabilità di osservare un cifrato* $c$ *è data dalla somma su tutti i possibili messaggi* $m' \in \mathcal{M}$*:*

$$\Pr[C = c] = \sum_{m' \in \mathcal{M}} \Pr[C = c \mid M = m'] \cdot \Pr[M = m'] \tag{6}$$

*Sostituendo il risultato del Passaggio 1,* $\Pr[C = c \mid M = m'] = 1/2^\ell$ *per ogni* $m'$ *(poiché esiste sempre un'unica chiave che mappa* $m'$ *a* $c$*):*

$$\Pr[C = c] = \sum_{m' \in \mathcal{M}} \frac{1}{2^\ell} \cdot \Pr[M = m'] = \frac{1}{2^\ell} \sum_{m' \in \mathcal{M}} \Pr[M = m'] \tag{7}$$

*Poiché la somma delle probabilità su tutti i possibili messaggi in* $\mathcal{M}$ *è 1 (*$\sum_{m' \in \mathcal{M}} \Pr[M = m'] = 1$*), otteniamo:*

$$\Pr[C = c] = \frac{1}{2^\ell} = \frac{1}{|\,\mathcal{C}\,|} \tag{8}$$

*Questo risultato, intuitivamente, dimostra che* ***il cifrato*** $c$ ***è distribuito uniformemente*** *su tutto lo spazio dei cifrati, indipendentemente dalla distribuzione del messaggio* $M$*.*

**Passaggio 3: Conclusione**

*Sostituendo i risultati dei Passaggi 1 e 2 nella formula di Bayes (4):*

$$\Pr[M = m \mid C = c] = \frac{\left(\frac{1}{2^\ell}\right) \cdot \Pr[M = m]}{\frac{1}{2^\ell}} = \Pr[M = m] \tag{9}$$

*La condizione di Segretezza Perfetta è soddisfatta. L'OTP è perfettamente segreto.*

# 15 Limitazioni e Teorema di Shannon

Il One-Time Pad è uno schema ideale, ma soffre di importanti limitazioni pratiche che rendono necessarie le **assunzioni computazionali** per gli schemi moderni.

## 15.1 Condizione Necessaria per la Segretezza Perfetta

**Teorema** (Condizione di Segretezza Perfetta)**.** *Un requisito* ***necessario*** *affinché uno schema di cifratura sia perfettamente segreto è che la dimensione dello spazio delle chiavi* $\mathcal{K}$ *sia* ***almeno pari*** *alla dimensione dello spazio dei messaggi* $\mathcal{M}$*:*

$$|\,\mathcal{K}\,| \geq |\,\mathcal{M}\,| \tag{10}$$

Nell'OTP, si ha esattamente $|\,\mathcal{K}\,| = |\,\mathcal{M}\,| = |\,\mathcal{C}\,|$. Se fosse $|\,\mathcal{K}\,| < |\,\mathcal{M}\,|$, esisterebbe almeno un cifrato $c$ a cui corrispondono più messaggi, violando l'uniformità e quindi la segretezza perfetta.

**Dimostrazione.** *Dimostriamo per assurdo che se* $|\,K\,| < |\,M\,|$*, lo schema non può essere perfettamente segreto. Sia dato un cifrato* $c$*. Definiamo l'insieme dei possibili messaggi in chiaro che potrebbero aver generato* $c$ *come* $\mathcal{M}(c) = \{m \in \mathcal{M} \mid \exists\, k \in \mathcal{K}$ *tale che* $m = Dec_k(c)\}$*. Chiaramente, il numero di possibili messaggi in* $\mathcal{M}(c)$ *non può essere maggiore del numero di chiavi, quindi* $|\,\mathcal{M}(c)\,| \leq |\,\mathcal{K}\,|$*. Se assumiamo* $|\,K\,| < |\,M\,|$*, allora per qualsiasi* $c$*, abbiamo* $|\,\mathcal{M}(c)\,| \leq |\,\mathcal{K}\,| < |\,\mathcal{M}\,|$*. Questo implica che esiste almeno un*

*messaggio $m_0 \in \mathcal{M}$ che non è in $\mathcal{M}(c)$. Per questo messaggio $m_0$, la probabilità a posteriori, dopo aver osservato $c$, è $Pr[M = m_0 \mid C = c] = 0$. Tuttavia, se scegliamo una distribuzione su $\mathcal{M}$ tale che $Pr[M = m_0] > 0$ (es. la distribuzione uniforme, dove $Pr[M = m_0] = 1/ \mid \mathcal{M} \mid$), allora $Pr[M = m_0 \mid C = c] \neq Pr[M = m_0]$. Questo viola la definizione di segretezza perfetta. Dunque, deve essere $\mid \mathcal{K} \mid \geq \mid \mathcal{M} \mid$.*

## 15.2 Uso Singolo della Chiave

Una limitazione critica, esemplificata dall'OTP ma intrinseca a tutti gli schemi perfettamente segreti, è che la chiave deve essere usata **una sola volta**. Se la stessa chiave $k$ viene usata per cifrare due messaggi $m$ e $m'$, un avversario può intercettare i due cifrati $c = m \oplus k$ e $c' = m' \oplus k$. Calcolando lo XOR tra i due cifrati, l'avversario ottiene:

$$c \oplus c' = (m \oplus k) \oplus (m' \oplus k) = m \oplus m'$$

Questa operazione elimina la chiave e rivela lo XOR dei due messaggi originali, una quantità enorme di informazione che può portare alla decifrazione di entrambi i messaggi tramite analisi statistica.

## 15.3 Teorema di Shannon (Characterization of Perfect Secrecy)

Il teorema di Shannon stabilisce le condizioni **necessarie e sufficienti** per la Segretezza Perfetta in schemi con spazi di messaggi, chiavi e cifrati della stessa dimensione.

**Teorema** (Teorema 2.11 di Shannon)**.** *Sia $(Gen, Enc, Dec)$ uno schema di cifratura con $\mid M \mid = \mid K \mid = \mid C \mid$. Lo schema è perfettamente segreto se e solo se:*

1. *La chiave $k \in \mathcal{K}$ è scelta in modo uniformemente casuale, i.e.,* $\Pr[K = k] = 1/ \mid \mathcal{K} \mid$.
2. *Per ogni messaggio $m \in \mathcal{M}$ e ogni cifrato $c \in \mathcal{C}$, esiste una e una sola chiave $k \in \mathcal{K}$ tale che $Enc_k(m) = c$.*

Il One-Time Pad soddisfa entrambe le condizioni.

Tabella 4: Riassunto della Sicurezza Segretezza Perfetta vs. Computazionale

| **Proprietà** | **Segretezza Perfetta (e.g., OTP)** | **Sicurezza Computazionale** |
|---|---|---|
| Potere dell'Avversario | Illimitato, illimitato (Informational) | Limitato, tempo polinomiale (Computational) |
| Assunzioni Richieste | Nessuna (solo def. e prove) | Difficoltà di problemi matematici (e.g., fattorizzazione) |
| Relazione Chiave/Messaggio | $\mid \mathcal{K} \mid \geq \mid \mathcal{M} \mid$ (chiave lunga come il messaggio) | $\mid \mathcal{K} \mid \ll \mid \mathcal{M} \mid$ (chiave corta, riutilizzabile) |
| Sicurezza | Incondizionata (perfetta) | Condizionale (asintotica/concreta) |

# 16 Lezione 4: Approcci alla Sicurezza Computazionale

Dopo aver analizzato i limiti intrinseci della Segretezza Perfetta (che richiede $\mid \mathcal{K} \mid \geq \mid \mathcal{M} \mid$), la crittografia moderna introduce il concetto di **Segretezza Computazionale** per superare tali restrizioni. Questo approccio si basa su due rilassamenti fondamentali rispetto alla Segretezza Perfetta:

1. **Limitazione del potere dell'Avversario (Adv):** Si assume che l'Adv abbia un potere computazionale limitato.
2. **Ammissione di una piccola probabilità di successo:** Si accetta che l'Adv possa avere successo nel "rompere" lo schema, ma solo con una probabilità molto piccola.

Esistono due approcci principali per formalizzare queste idee.

## 16.1 Approccio Concreto

L'approccio concreto definisce la sicurezza quantificando esplicitamente le risorse dell'avversario e la sua probabilità di successo.

- **Definizione:** Uno schema è **(t, $\epsilon$)-sicuro** se qualsiasi avversario che esegue per un tempo al più $t$ ha successo nel rompere lo schema con una probabilità al più $\epsilon$.
- **Esempio:** Per uno schema con chiavi di $n$ bit, si desidera che la sicurezza si avvicini a quella di un attacco di forza bruta. La probabilità di successo per un avversario che esegue per un tempo $t$ dovrebbe essere circa $\epsilon \approx \frac{c \cdot t}{2^n}$ per una piccola costante $c$. Per una chiave a 128 bit ($n = 128$), oggi si considerano valori come $t = 2^{80}$ operazioni e $\epsilon = 2^{-60}$.

- **Svantaggi:** Questo approccio rende difficile la comparazione tra schemi e la composizione di protocolli. Inoltre, i parametri $(t, \epsilon)$ diventano rapidamente obsoleti con l'avanzare della tecnologia (es. Legge di Moore).

## 16.2 Approccio Asintotico

Questo è l'approccio dominante nella crittografia teorica e lega la sicurezza a un **parametro di sicurezza** $n$ (tipicamente la lunghezza della chiave).

## 16.3 Perché i Computer più Veloci Favoriscono le Parti Oneste

Un'importante conseguenza dell'approccio asintotico è che il progresso tecnologico (es. computer più veloci) gioca a favore dei difensori, non degli attaccanti. Consideriamo un protocollo dove le parti oneste impiegano $10^6 \cdot n^2$ cicli di CPU e un avversario impiega $10^8 \cdot n^4$ cicli per avere successo con probabilità $2^{-n/2}$.

- **Oggi (CPU a 2GHz):** Con $n = 80$, le parti oneste impiegano 3.2 secondi. L'avversario impiega circa 3 settimane per un attacco con probabilità di successo $2^{-40}$.
- **In futuro (CPU a 8GHz, 4 volte più veloce):** Le parti oneste possono aumentare il parametro di sicurezza a $n = 160$ e impiegare comunque 3.2 secondi per le loro operazioni. L'avversario, nonostante la CPU più veloce, ora necessita di più di 13 settimane per un attacco la cui probabilità di successo è scesa a $2^{-80}$.

Incrementando $n$, le parti oneste mantengono costante il loro tempo di esecuzione, mentre il lavoro dell'avversario diventa molto più difficile.

## 16.4 Segretezza Computazionale e Pseudocasualità

Dopo aver analizzato i limiti intrinseci della **Segretezza Perfetta** (che richiede $| K | \geq | M |$, come visto nell'OTP), la crittografia moderna introduce il concetto di **Segretezza Computazionale** per superare tali restrizioni, permettendo di cifrare messaggi di lunghezza arbitraria con una chiave di dimensione fissa (e.g., 128 bit per molti GigaByte di dati).

Questo approccio si basa su due rilassamenti fondamentali rispetto alla Segretezza Perfetta:

1. **Limitazione del potere dell'Avversario (Adv):** Si assume che l'Adv abbia un potere limitato, in particolare che sia un algoritmo **PPT** (**P**robabilistico **P**olinomiale in **T**empo).

2. **Ammissione di una piccola probabilità di successo:** Si accetta che l'Adv possa avere successo nel "rompere" lo schema, ma solo con una probabilità **trascurabile**.

## 16.5 Algoritmi PPT e Parametro di Sicurezza

Tutta la teoria crittografica moderna si basa sulla nozione di **efficienza** misurata rispetto a un **parametro di sicurezza** $n$.

**Definizione** (Algoritmo PPT)**.** *Un algoritmo $A$ è **Probabilistico Polinomiale in Tempo (PPT)** se:*

1. *È **probabilistico**: utilizza dei* random coin *(bit casuali) nel suo processo di calcolo.*
2. *Ha un tempo di esecuzione limitato da un polinomio $p(\cdot)$: per ogni input di lunghezza $n$, l'algoritmo termina in tempo $T(n) \leq p(n)$, dove $p$ è un polinomio fissato.*

In questo modello, gli Avversari ($A$) e gli algoritmi dello schema crittografico (Gen, Enc, Dec) devono essere PPT.

## 16.6 Funzioni Trascurabili

Una probabilità di successo è considerata accettabile se è **trascurabile**, ovvero se decresce più velocemente di qualsiasi inversa di un polinomio. Questa è la formalizzazione della condizione "piccola probabilità di successo".

**Definizione** (Funzione Trascurabile)**.** *Una funzione $negl : \mathbb{N} \to \mathbb{R}^{\geq 0}$ è detta **trascurabile** se per ogni polinomio positivo $p(\cdot)$, esiste un intero $N$ tale che per tutti gli $n > N$, si ha:*

$$negl(n) < \frac{1}{p(n)} \tag{11}$$

Intuitivamente, se la probabilità di successo di un Adv è data da negl($n$), tale probabilità è così piccola che non sarà mai un problema pratico, poiché è dominata dalla funzione $\frac{1}{p(n)}$ per ogni $p(n)$.

**Teorema** (Proprietà di Chiusura delle Funzioni Trascurabili)**.** *Siano $negl_1(n)$ e $negl_2(n)$ due funzioni trascurabili, e sia $p(n)$ un polinomio. Allora:*

1. *La somma di due funzioni trascurabili è trascurabile: $negl_1(n) + negl_2(n)$ è trascurabile.*
2. *La moltiplicazione di una funzione trascurabile per un polinomio è trascurabile: $p(n) \cdot negl_1(n)$ è trascurabile.*

Queste proprietà sono cruciali nella crittografia per dimostrare la sicurezza tramite **riduzione**. Se un attacco richiede un numero polinomiale di operazioni e ogni singola operazione ha una probabilità di successo trascurabile, la probabilità totale di successo rimane trascurabile.

### Esempi di Funzioni Trascurabili

Esempi di funzioni che soddisfano la condizione di trascurabilità sono:

- L'esponenziale negativa di $n$: $\text{negl}(n) = 2^{-n}$ (o $2^{-cn}$ per $c > 0$).
- Funzioni che crescono più lentamente dell'esponenziale ma più velocemente di qualsiasi polinomio inverso:

$$\text{negl}(n) = n^{-\log(n)} \quad \text{oppure} \quad \text{negl}(n) = 2^{-\sqrt{n}}$$

**Non-Esempi (Funzioni non trascurabili):**

- Qualsiasi funzione che sia il reciproco di un polinomio, ad esempio: $\text{negl}(n) = \frac{1}{n}$, $\text{negl}(n) = \frac{1}{n^{100}}$.

## 16.7 Il Concetto di Pseudocasualità

La Segretezza Computazionale è strettamente legata alla nozione di **Pseudocasualità** (*Pseudorandomness*).

Un oggetto (come la sequenza di bit generata da una chiave) è detto **pseudocasuale** se, pur non essendo statisticamente casuale, appare tale a qualsiasi algoritmo PPT. In altre parole, nessun algoritmo PPT è in grado di **distinguere** una sequenza pseudocasuale da una genuinamente casuale con probabilità non trascurabile. Questo concetto è centrale per la costruzione di tutti gli schemi crittografici computazionalmente sicuri.

## 16.8 La Necessità di Entrambi i Rilassamenti

La sicurezza computazionale si basa su due rilassamenti: limitare il **tempo** dell'avversario e ammettere una **piccola probabilità di successo**. Entrambi sono strettamente necessari.

- **Perché limitare il tempo non basta?** Se ammettessimo una probabilità di successo pari a zero ma permettessimo un tempo illimitato, un avversario potrebbe semplicemente provare tutte le chiavi (ricerca esaustiva), trovando quella corretta con probabilità 1. Dobbiamo quindi escludere tali ricerche limitando il tempo di esecuzione dell'avversario a polinomiale.

- **Perché ammettere una probabilità di successo non basta?** Se limitassimo solo il tempo dell'avversario a polinomiale, egli potrebbe comunque eseguire un attacco molto semplice: scegliere una chiave $k'$ a caso e verificare se decifra correttamente una coppia messaggio/cifrato nota. Questo attacco richiede tempo costante e ha successo con probabilità $1/ \mid \mathcal{K} \mid$. Poiché $\mid \mathcal{K} \mid$ è al più polinomiale in $n$, questa probabilità non è trascurabile. Dobbiamo quindi richiedere che la probabilità di successo sia trascurabile per escludere anche questi attacchi "fortunati".

# 17 Lezione 5: Schemi Computazionalmente Sicuri e Sicurezza Semantica

La Segretezza Computazionale viene formalizzata attraverso la definizione di uno **schema di cifratura computazionalmente sicuro** e, in particolare, attraverso il criterio di **Indistinguibilità**.

## 17.1 Definizione di Schema di Cifratura a Chiave Privata Computazionale

Uno schema di cifratura non è più richiesto di essere perfettamente corretto o perfettamente sicuro, ma deve essere **PPT-implementabile**.

**Definizione** (Schema di Cifratura a Chiave Privata)**.** *Uno schema di cifratura a chiave privata è una tripla di algoritmi PPT* $\Pi = (Gen, Enc, Dec)$ *tale che:*

1. ***Generazione della chiave (Gen):*** $Gen(1^n) \rightarrow k$*, algoritmo* ***probabilistico*** *che genera una chiave* $k \in K$*, con* $\mid k \mid \geq n$ *(parametro di sicurezza).*
2. ***Cifratura (Enc):*** $c \leftarrow Enc_k(m)$*, algoritmo* ***probabilistico*** *che, data la chiave* $k$ *e il messaggio* $m \in \{0,1\}^*$*, produce il cifrato* $c \in \{0,1\}^*$*.*
3. ***Decifratura (Dec):*** $m := Dec_k(c)$*, algoritmo* ***deterministico*** *che, data la chiave* $k$ *e il cifrato* $c$*, produce il messaggio* $m$ *(o* $\perp$ *in caso di errore).*

**Definizione** (Correttezza)**.** *Uno schema* $\Pi = (Gen, Enc, Dec)$ *soddisfa la correttezza se, per ogni parametro di sicurezza* $n$*, per ogni chiave* $k \leftarrow Gen(1^n)$ *e per ogni messaggio* $m$ *nell'insieme dei messaggi, la probabilità che il decifrato sia il messaggio originale è 1:*

$$\Pr_{k \leftarrow Gen(1^n)}[Dec_k(Enc_k(m)) = m] = 1 \tag{12}$$

Nei sistemi reali, si ammette che la probabilità di errore sia al più trascurabile, ma per scopi didattici la correttezza è spesso richiesta perfetta (probabilità 1).

## 17.2 Indistinguibilità e Sicurezza Contro Eavesdropper

La sicurezza computazionale è formalizzata attraverso l'esperimento di **Indistinguibilità** in presenza di un **eavesdropper** (ascoltatore passivo), noto anche come sicurezza **IND-CPA-0** o **EAV-Security**.

**Definizione** (Esperimento di Indistinguibilità $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n)$)**.** *Per uno schema* $\Pi = (Gen, Enc, Dec)$ *e un Adv A PPT, si definisce l'esperimento come segue:*

1. *L'Adv* $A(1^n)$ *sceglie una coppia di messaggi* $m_0, m_1$ *di uguale lunghezza (*$\mid m_0 \mid = \mid m_1 \mid$*).*
2. *Una chiave* $k$ *è generata casualmente:* $k \leftarrow Gen(1^n)$*.*
3. *Un bit casuale* $b$ *è scelto:* $b \leftarrow \{0,1\}$*.*
4. *Il cifrato* $c$ *è calcolato come* $Enc_k(m_b)$*.*
5. *L'Adv A riceve* $c$ *e restituisce un bit di indovinello* $b'$*.*
6. *L'output dell'esperimento è 1 se* $b' = b$ *(Adv ha successo), ed è 0 altrimenti.*

**Definizione** (Sicurezza Computazionale (Indistinguibilità))**.** *Uno schema di cifratura a chiave privata* $\Pi$ *è* ***computazionalmente sicuro in presenza di un eavesdropper*** *se per ogni Adv A PPT, la probabilità che A abbia successo nell'esperimento di indistinguibilità* $PrivK^{eav}_{A,\Pi}(n)$ *è al più* $1/2$ *più una funzione trascurabile. Formalmente:*

$$\Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + negl(n) \tag{13}$$

*dove* $negl(n)$ *è una funzione trascurabile.*

Il successo di $A$ è misurato dalla differenza tra la probabilità che $A$ indovini correttamente e la probabilità di indovinare a caso $(1/2)$. Questa differenza deve essere trascurabile:

$$\left| \Pr[\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n) = 1] - \frac{1}{2} \right| \leq \mathrm{negl}(n)$$

## 17.3 Esempio di Riduzione: Indistinguibilità implica Imprevedibilità

Un teorema fondamentale mostra che se uno schema è EAV-sicuro, allora nessun avversario PPT può predire un qualsiasi singolo bit $m_i$ del messaggio in chiaro $m$ con probabilità significativamente migliore di $1/2$.

**Teorema** (Teorema 3.10)**.** *Sia* $\Pi = (Enc, Dec)$ *uno schema di cifratura a lunghezza fissa* $l$ *che sia EAV-sicuro. Allora, per ogni avversario A a tempo polinomiale probabilistico (PPT) e per ogni posizione* $i \in \{1, \ldots, l\}$*, esiste una funzione trascurabile negl tale che:*

$$Pr[A(1^n, Enc_k(m)) = m_i] \leq \frac{1}{2} + negl(n)$$

*dove la probabilità è calcolata sulla scelta uniforme di* $m \in \{0,1\}^l$, $k \in \{0,1\}^n$, *e sulla casualità di A e Enc.*

**Dimostrazione** (Dimostrazione (per Riduzione))**.** *Supponiamo per assurdo che esista un avversario A PPT e un polinomio* $p(\cdot)$ *tale che, per infiniti valori di n:*

$$Pr[A(1^n, Enc_k(m)) = m_i] \geq \frac{1}{2} + \frac{1}{p(n)}$$

*Sia* $\epsilon(n) = 1/p(n)$*. Costruiamo un avversario A' che usa A come subroutine per rompere la sicurezza EAV di* $\Pi$*.*

***Costruzione di A'****:*

1. *A' sceglie due messaggi* $m_0, m_1 \in \{0,1\}^l$ *tali che differiscano solo nell'i-esimo bit (cioè* $(m_0)_i = 0$ *e* $(m_1)_i = 1$*). Invia* $(m_0, m_1)$ *al suo challenger.*
2. *A' riceve un cifrato di sfida* $c \leftarrow Enc_k(m_b)$*, dove* $b \in \{0,1\}$ *è casuale.*
3. *A' invoca A su c. Sia* $b'$ *l'output di* $A(1^n, c)$*.*
4. *A' restituisce* $b'$ *come sua ipotesi per b.*

***Analisi del Successo di A'****: La probabilità che A' vinca è la probabilità che* $b' = b$*.*

$$\begin{aligned} Pr[PrivK^{eav}_{A',\Pi}(n) = 1] &= \frac{1}{2} \cdot Pr[b' = 0 \mid b = 0] + \frac{1}{2} \cdot Pr[b' = 1 \mid b = 1] \\ &= \frac{1}{2} \cdot Pr[A(Enc_k(m_0)) = 0] + \frac{1}{2} \cdot Pr[A(Enc_k(m_1)) = 1] \end{aligned}$$

*Questa espressione è esattamente la probabilità di successo di A nel predire il bit* $m_i$ *quando m è scelto uniformemente. Sia* $I_0 = \{m \in \{0,1\}^l \mid m_i = 0\}$ *e* $I_1 = \{m \in \{0,1\}^l \mid m_i = 1\}$

$$Pr[A(1^n, Enc_k(m)) = m_i] = \frac{1}{2} \cdot Pr_{m_0 \leftarrow I_0}[A(Enc_k(m_0)) = 0] + \frac{1}{2} \cdot Pr_{m_1 \leftarrow I_1}[A(Enc_k(m_1)) = 1]$$

*Quindi, abbiamo*

$$Pr[PrivK^{eav}_{A',\Pi}(n) = 1] = Pr[A(1^n, Enc_k(m)) = m_i]$$

***Contraddizione****: Dalla nostra assunzione per assurdo, segue che:*

$$Pr[PrivK^{eav}_{A',\Pi}(n) = 1] \geq \frac{1}{2} + \epsilon(n)$$

*Questo significa che A' ha un vantaggio non trascurabile* $\epsilon(n)$ *nel vincere l'esperimento di indistinguibilità, il che contraddice l'ipotesi che* $\Pi$ *sia EAV-sicuro. L'assunzione iniziale deve quindi essere falsa.*

# 18 Sicurezza basata sull'Indistinguibilità (IND-EAV)

L'approccio basato sull'indistinguibilità è lo standard pratico per definire la sicurezza computazionale.

## Esperimento di Sicurezza $\mathbf{PrivK}^{\mathbf{eav}}_{\mathcal{A},\Pi}(n)$

Sia $\Pi = (Gen, Enc, Dec)$ uno schema di cifratura e $\mathcal{A}$ un avversario PPT (Probabilistico a Tempo Polinomiale). L'esperimento procede come segue:

1. **Setup**: Il Challenger $\mathcal{C}$ esegue $k \leftarrow Gen(1^n)$ e lo tiene segreto.
2. **Sfida (Challenge)**: $\mathcal{A}$ sceglie e invia a $\mathcal{C}$ due messaggi $m_0, m_1$ di uguale lunghezza ($\mid m_0 \mid = \mid m_1 \mid$).
3. **Risposta**: $\mathcal{C}$ sceglie un bit casuale $b \in \{0,1\}$ e calcola il cifrato di sfida $c \leftarrow Enc_k(m_b)$. $\mathcal{C}$ invia $c$ ad $\mathcal{A}$.
4. **Indovinello**: $\mathcal{A}$ restituisce un bit $b' \in \{0,1\}$.
5. **Esito**: L'output dell'esperimento è 1 se $b' = b$ (l'avversario ha successo), 0 altrimenti.

## Definizione di Sicurezza Computazionale (IND-EAV)

**Definizione.** *Uno schema di cifratura $\Pi = (Gen, Enc, Dec)$ è **computazionalmente indistinguibile** in presenza di un **eavesdropper** (IND-EAV) se, per ogni avversario $\mathcal{A}$ PPT, la probabilità che $\mathcal{A}$ vinca l'esperimento è appena superiore a quella di indovinare a caso:*

$$Pr[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

*dove $negl(n)$ è una funzione trascurabile.*

**Formulazione Equivalente dell'Indistinguibilità.** Una definizione equivalente, talvolta più comoda nelle dimostrazioni, afferma che per ogni avversario A PPT, la sua probabilità di output '1' (o qualsiasi altro valore fisso) quando il messaggio di sfida è una cifratura di $m_0$ deve essere quasi identica alla sua probabilità di output '1' quando è una cifratura di $m_1$. Formalmente, per una funzione trascurabile negl:

$$\mid Pr[\text{out}_A(\text{PrivK}^{\text{eav}}_{A,\Pi}(n,0)) = 1] - Pr[\text{out}_A(\text{PrivK}^{\text{eav}}_{A,\Pi}(n,1)) = 1] \mid \leq \text{negl}(n)$$

dove $\text{PrivK}^{\text{eav}}_{A,\Pi}(n,b)$ è l'esperimento in cui il bit di sfida è fissato a $b$.

**Nota sulla Lunghezza del Messaggio.** È importante notare che questa definizione di sicurezza non richiede allo schema di nascondere la lunghezza del messaggio cifrato. Infatti, l'avversario A deve scegliere due messaggi $m_0, m_1$ della **stessa lunghezza** [464]. Se le lunghezze fossero diverse, distinguerli potrebbe essere banale. Nei casi pratici in cui anche la lunghezza del messaggio è un'informazione sensibile, è necessario usare tecniche di *padding* per estendere tutti i messaggi a una lunghezza fissa standard.

## 18.1 Sicurezza Semantica

La **Sicurezza Semantica** è una definizione alternativa e concettualmente più forte, che cattura l'idea informale che il cifrato non deve rivelare **alcuna informazione aggiuntiva** sul messaggio sottostante, anche se l'Adv possiede informazioni ausiliarie.

**Definizione** (Sicurezza Semantica)**.** *Uno schema di cifratura a chiave privata* $\Pi$ *è* ***semanticamente sicuro in presenza di un eavesdropper*** *se, per ogni Adv A PPT, esiste un simulatore* $A'$ *(un altro Adv PPT) tale che, per qualsiasi distribuzione di probabilità sui messaggi (campionabile tramite* $Samp(1^n)$*) e per ogni coppia di funzioni* $f$ *e* $h$ *calcolabili in tempo polinomiale, la seguente disuguaglianza vale per una funzione trascurabile* $negl(n)$*:*

$$| \Pr[A(1^n, Enc_k(m), h(m)) = f(m)] - \Pr[A'(1^n, | m |, h(m)) = f(m)] | \leq negl(n) \quad (14)$$

*dove* $m$ *è il messaggio scelto secondo* $Samp(1^n)$ *e la chiave* $k$ *è scelta uniformemente.*

In termini semplici: qualsiasi cosa $A$ possa calcolare su $m$ usando il cifrato $\text{Enc}_k(m)$ (il risultato $f(m)$), il simulatore $A'$ deve essere in grado di calcolarlo con la stessa probabilità di successo ($\pm\text{negl}(n)$) utilizzando **solo la lunghezza di** $m$ ($| m |$) e le informazioni ausiliarie $h(m)$, ma **senza** il cifrato $c$. Questo dimostra che il cifrato non fornisce un reale vantaggio informativo.

### Equivalenza tra Sicurezza Semantica e Indistinguibilità

Un risultato fondamentale della crittografia moderna stabilisce che la **Sicurezza Semantica è equivalente alla Sicurezza basata sull'Indistinguibilità** (Formula 13).

**Teorema.** *Uno schema di cifratura a chiave privata* $\Pi = (Gen, Enc, Dec)$ *è* ***semanticamente sicuro*** *se e solo se soddisfa il criterio di* ***indistinguibilità*** *(EAV-Security).*

Questa equivalenza è di grande importanza pratica, poiché la dimostrazione della sicurezza tramite l'esperimento di Indistinguibilità è molto più gestibile e diretta rispetto alla prova formale della Sicurezza Semantica.

Certamente, ecco gli appunti per la lezione 6 sulla costruzione di schemi di cifratura sicuri, redatti nello stile del documento fornito e integrando le informazioni dalla trascrizione, dalle slide e dal libro di testo.

# 19 Lezione 6: Costruzioni Sicure e Generatori Pseudocasuali

Nelle lezioni precedenti abbiamo definito rigorosamente cosa si intende per sicurezza computazionale, in particolare attraverso la nozione di indistinguibilità (IND-EAV). Ora ci concentriamo sulla domanda fondamentale: esistono schemi di cifratura che soddisfano questa definizione? E come possiamo costruirli?.

Questa lezione introduce una delle primitive crittografiche più importanti, il generatore di numeri pseudocasuali (PRG), e mostra come esso possa essere usato per costruire uno schema di cifratura computazionalmente sicuro, superando i limiti della segretezza perfetta. Vedremo anche la tecnica di dimostrazione per eccellenza della crittografia moderna: la prova per riduzione.

# 20 Generatori Pseudocasuali (PRG)

Un generatore pseudocasuale (in inglese, *Pseudorandom Generator* o PRG) è il primo blocco di base fondamentale che incontriamo per la costruzione di schemi sicuri.

**Definizione.** *(Generatore Pseudocasuale - Informale) Un generatore pseudocasuale $G$ è un algoritmo deterministico ed efficiente che prende in input una stringa corta e uniforme, chiamata seme (*seed*), e la trasforma in una stringa più lunga che "sembra" uniforme a un osservatore computazionalmente limitato.*

L'idea di "sembrare uniforme" è cruciale e deve essere formalizzata. Storicamente, i generatori di numeri casuali (studiati sin dagli anni '40) erano progettati per superare specifici test statistici *ad hoc*. Ad esempio, si verificava che la frequenza di 0 e 1 fosse bilanciata, o che la parità dei bit fosse distribuita uniformemente.

L'approccio crittografico introdotto negli anni '80 è molto più stringente: un buon generatore pseudocasuale deve superare tutti i test statistici efficienti. In altre parole, nessun algoritmo PPT deve essere in grado di distinguere l'output del generatore da una stringa genuinamente casuale della stessa lunghezza.

## 20.1 Definizione Formale di PRG

Per formalizzare questo concetto, introduciamo il concetto di indistinguibilità tra due ensemble di distribuzioni di probabilità.

Sia $U_k$ la distribuzione uniforme su stringhe di lunghezza $k$. Sia $G$ un algoritmo deterministico che, su input un seme $s$ di lunghezza $n$, produce un output di lunghezza $l(n)$. La distribuzione indotta da $G$ è $\{G(s)\}_{s \leftarrow U_n}$.

**Definizione.** *(Generatore Pseudocasuale - Formale) Sia $l(n)$ un polinomio e $G$ un algoritmo deterministico a tempo polinomiale tale che, per ogni $n$ e per ogni seme $s \in$*

*$\{0,1\}^n$, l'output $G(s)$ è una stringa di lunghezza $l(n)$ bit. Diremo che $G$ è un generatore pseudocasuale (PRG) se valgono le seguenti condizioni:*

1. ***Espansione (Expansion):*** *Per ogni $n$, risulta $l(n) > n$. La lunghezza dell'output deve essere strettamente maggiore di quella del seme.*
2. ***Pseudocasualità (Pseudorandomness):*** *Per ogni algoritmo distinguisher $D$ PPT, esiste una funzione trascurabile $negl(n)$ tale che:*

$$| \Pr_{s \leftarrow U_n}[D(G(s)) = 1] - \Pr_{r \leftarrow U_{l(n)}}[D(r) = 1] | \leq negl(n)$$

*dove la prima probabilità è calcolata sulla scelta uniforme del seme $s$ e sulla casualità interna di $D$, mentre la seconda è calcolata sulla scelta uniforme della stringa $r$ e sulla casualità di $D$.*

## 20.2 Esempio di Costruzione Non Sicura

Per comprendere meglio la definizione, analizziamo un generatore che non è un PRG.

**Esempio 1 (Costruzione non sicura)** Consideriamo l'algoritmo $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ definito come segue:

$$G(s) = s \ || \ \bigoplus_{i=1}^{n} s_i$$

dove $s = s_1 ... s_n$ e $||$ denota la concatenazione. Questo generatore concatena al seme $s$ il bit di parità dei suoi bit. La condizione di espansione è soddisfatta ($l(n) = n + 1 > n$).

Tuttavia, non soddisfa la proprietà di pseudocasualità. Per dimostrarlo, è sufficiente esibire un *distinguisher* $D$ efficiente che nega la definizione.

**Distinguisher D:** Su input una stringa $w$ di lunghezza $n + 1$, $D$ calcola il bit di parità dei primi $n$ bit di $w$ e lo confronta con l'ultimo bit di $w$. Se sono uguali, $D$ restituisce 1, altrimenti 0.

$$D(w_1 ... w_{n+1}) = \begin{cases} 1 & \text{se } w_{n+1} = \bigoplus_{i=1}^{n} w_i \\ 0 & \text{altrimenti} \end{cases}$$

**Analisi:**

1. **Input Pseudocasuale:** Se l'input di $D$ è $w = G(s)$, per costruzione l'ultimo bit è sempre uguale alla parità dei primi $n$ bit. Quindi:

$$\Pr_{s \leftarrow U_n}[D(G(s)) = 1] = 1$$

2. **Input Casuale:** Se l'input di $D$ è una stringa uniforme $r \in \{0,1\}^{n+1}$, ogni bit di $r$ è 0 o 1 con probabilità $1/2$, indipendentemente dagli altri. Il valore di $\bigoplus_{i=1}^{n} r_i$ sarà 0 o 1, e l'ultimo bit $r_{n+1}$ corrisponderà a questo valore con probabilità $1/2$. Quindi:

$$\Pr_{r \leftarrow U_{n+1}}[D(r) = 1] = \frac{1}{2}$$

La differenza tra le probabilità è $| 1 - 1/2 | = 1/2$, che è una costante e quindi una funzione non trascurabile. Di conseguenza, $G$ non è un generatore pseudocasuale.

### 20.3 Osservazioni sui PRG

- **La distribuzione di un PRG è lontana dall'essere uniforme:** Un PRG $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ può generare al più $2^n$ stringhe distinte, dato che ci sono solo $2^n$ semi possibili. Lo spazio di tutte le possibili stringhe di output è $2^{l(n)}$. La frazione di stringhe che $G$ può generare è $\frac{2^n}{2^{l(n)}} = \frac{1}{2^{l(n)-n}}$, che è esponenzialmente piccola. Questo significa che la stragrande maggioranza delle stringhe non può essere generata.

- **La sicurezza dipende dal potere computazionale limitato:** Un avversario con potere illimitato può sempre rompere un PRG. Potrebbe, dato un output $w$, calcolare $G(s)$ per tutti i $2^n$ semi possibili e verificare se $w$ appartiene all'immagine di $G$. Questo attacco richiede tempo esponenziale e non è quindi una minaccia nel modello computazionale.

- **Esistenza di PRG:** L'esistenza di PRG non è stata provata incondizionatamente. Tuttavia, si può dimostrare che se esistono le **funzioni one-way**, allora esistono anche i PRG. In pratica, si usano buone costruzioni basate su cifrari a flusso (stream ciphers).

## 21 Prove per Riduzione

La tecnica della prova per riduzione è la metodologia standard per dimostrare la sicurezza di una costruzione crittografica. L'idea è di mostrare che un attacco efficace contro la nostra costruzione implicherebbe la capacità di risolvere un problema sottostante che si assume essere difficile, o di rompere una primitiva che si assume essere sicura.

**Struttura di una Riduzione:** Supponiamo di voler dimostrare che una costruzione $\Pi$ è sicura, basandoci sull'assunzione che una primitiva $X$ (es. un PRG) sia sicura. La dimostrazione procede per assurdo:

1. **Assunzione di base:** La primitiva $X$ è sicura.
2. **Ipotesi per assurdo:** La costruzione $\Pi$ non è sicura.
3. **Conseguenza:** Se $\Pi$ non è sicura, allora esiste un avversario $\mathcal{A}$ PPT che "rompe" $\Pi$ con probabilità non trascurabile $\epsilon(n)$.
4. **Costruzione della Riduzione:** Si costruisce un nuovo algoritmo $\mathcal{A}'$ (la riduzione) che ha l'obiettivo di rompere la primitiva $X$. $\mathcal{A}'$ usa $\mathcal{A}$ come una subroutine "black-box".
5. **Simulazione:** $\mathcal{A}'$ simula per $\mathcal{A}$ l'ambiente di attacco a $\Pi$ in modo che, dal punto di vista di $\mathcal{A}$, l'interazione sia indistinguibile da un attacco reale a $\Pi$.

6. **Collegamento:** $\mathcal{A}'$ sfrutta l'output di $\mathcal{A}$ per risolvere il problema difficile o rompere la primitiva $X$.

7. **Contraddizione:** Poiché $\mathcal{A}$ ha successo con probabilità non trascurabile, anche $\mathcal{A}'$ avrà successo nel rompere $X$ con probabilità non trascurabile. Questo contraddice l'assunzione di base (punto 1) che $X$ sia sicura.

8. **Conclusione:** L'ipotesi per assurdo (punto 2) deve essere falsa. Pertanto, la costruzione $\Pi$ deve essere sicura.

# 22 Costruzione di un Cifrario Sicuro da un PRG

Applichiamo ora la tecnica della riduzione per dimostrare la sicurezza di uno schema di cifratura costruito a partire da un generatore pseudocasuale. L'idea è una generalizzazione computazionale del One-Time Pad: invece di usare una chiave casuale lunga quanto il messaggio, usiamo una stringa pseudocasuale generata da un seme corto.

**Definizione.** *(Costruzione 3.17) Sia $G$ un PRG con fattore di espansione $l(n)$. Definiamo uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ per messaggi di lunghezza $l(n)$ come segue:*

- ***Gen($1^n$)**: Sceglie uniformemente a caso un seme $k \in \{0,1\}^n$ e lo restituisce come chiave.*
- ***Enc$_k$(m)**: Dato un messaggio $m \in \{0,1\}^{l(n)}$, calcola il pad pseudocasuale $G(k)$ e restituisce il cifrato $c := G(k) \oplus m$.*
- ***Dec$_k$(c)**: Dato un cifrato $c \in \{0,1\}^{l(n)}$, calcola $G(k)$ e restituisce il messaggio $m := G(k) \oplus c$.*

)

La correttezza è immediata, poiché $(G(k) \oplus m) \oplus G(k) = m$. L'enorme vantaggio di questo schema è che una chiave corta di $n$ bit (es. 128 bit) può essere usata per cifrare un messaggio molto lungo di $l(n)$ bit (es. 1 GigaByte).

**Teorema.** *(Teorema 3.16) Se $G$ è un generatore pseudocasuale, la Costruzione 3.17 è uno schema di cifratura a chiave privata a lunghezza fissa con cifrati indistinguibili in presenza di un eavesdropper (IND-EAV sicuro).*

**Dimostrazione.** *La prova segue la struttura della riduzione. Assumiamo per assurdo che la costruzione $\Pi$ non sia IND-EAV sicura. Allora esiste un avversario $\mathcal{A}$ PPT che vince l'esperimento $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ con probabilità non trascurabilmente maggiore di 1/2.*

$$\Pr[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \epsilon(n)$$

*dove $\epsilon(n)$ è una funzione non trascurabile.*

*Costruiamo un* distinguisher $\mathcal{D}$ *che tenta di rompere la pseudocasualità di* $G$. $\mathcal{D}$ *riceve in input una stringa* $\omega \in \{0,1\}^{l(n)}$ *e deve decidere se* $\omega$ *è l'output di* $G$ *su un seme casuale o se è una stringa uniformemente casuale.*

***Algoritmo del Distinguisher*** $\mathcal{D}^{\mathcal{A}}(\omega)$***:***

1. *Esegue* $\mathcal{A}(1^n)$ *per ottenere una coppia di messaggi* $m_0, m_1 \in \{0,1\}^{l(n)}$.
2. *Sceglie un bit* $b \in \{0,1\}$ *uniformemente a caso.*
3. *Calcola un cifrato di sfida* $c := \omega \oplus m_b$.
4. *Fornisce* $c$ *ad* $\mathcal{A}$ *e riceve in risposta il bit* $b'$.
5. *Se* $b' = b$, $\mathcal{D}$ *restituisce 1 (indovinando che* $\omega$ *è pseudocasuale); altrimenti, restituisce 0.*

***Analisi del comportamento di*** $\mathcal{D}$***:*** *Analizziamo la probabilità che* $\mathcal{D}$ *restituisca 1 in due casi.*

- ***Caso 1:*** $\omega$ ***è una stringa uniforme.*** *Se* $\omega$ *è una stringa* $r$ *scelta uniformemente a caso da* $\{0,1\}^{l(n)}$*, allora il cifrato* $c = r \oplus m_b$ *è una cifratura del messaggio* $m_b$ *secondo lo schema One-Time Pad. Poiché l'OTP è perfettamente segreto, la probabilità che* $\mathcal{A}$ *indovini correttamente* $b$ *è esattamente 1/2. Di conseguenza, la probabilità che* $\mathcal{D}$ *restituisca 1 è:*

$$\Pr_{r \leftarrow U_{l(n)}}[\mathcal{D}(r) = 1] = \Pr[PrivK^{eav}_{\mathcal{A},OTP}(n) = 1] = \frac{1}{2}$$

- ***Caso 2:*** $\omega$ ***è una stringa pseudocasuale.*** *Se* $\omega = G(k)$ *per un seme* $k$ *scelto uniformemente a caso, allora il cifrato* $c = G(k) \oplus m_b$ *è una cifratura di* $m_b$ *secondo la nostra costruzione* $\Pi$*. La probabilità che* $\mathcal{D}$ *restituisca 1 è la probabilità che* $\mathcal{A}$ *vinca l'esperimento di indistinguibilità per* $\Pi$*:*

$$\Pr_{k \leftarrow U_n}[\mathcal{D}(G(k)) = 1] = \Pr[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \epsilon(n)$$

*La differenza di probabilità con cui* $\mathcal{D}$ *distingue i due casi è:*

$$|\Pr[\mathcal{D}(G(k)) = 1] - \Pr[\mathcal{D}(r) = 1]| = |(\frac{1}{2} + \epsilon(n)) - \frac{1}{2}| = \epsilon(n)$$

*Poiché per ipotesi* $\epsilon(n)$ *è non trascurabile, abbiamo costruito un distinguisher* $\mathcal{D}$ *che rompe la pseudocasualità di* $G$ *con probabilità non trascurabile. Questo contraddice l'assunzione che* $G$ *sia un PRG. Pertanto, l'ipotesi iniziale che* $\Pi$ *non sia sicuro deve essere falsa.*

# 23 Lezione 7: Nozioni di Sicurezza più Forti

Nelle lezioni precedenti, abbiamo definito la sicurezza computazionale (IND-EAV) considerando un avversario passivo che osserva la cifratura di un singolo messaggio . Tuttavia, le applicazioni del mondo reale richiedono garanzie di sicurezza più robuste per scenari di attacco più potenti . In questa lezione, introduciamo nozioni di sicurezza più forti per affrontare due scenari più realistici: la sicurezza quando più messaggi vengono cifrati con la stessa chiave e la sicurezza contro avversari che possono influenzare i testi in chiaro da cifrare (attacchi con testo in chiaro scelto).

## 23.1 Sicurezza per Cifrature Multiple (IND-EAV-MULT)

In molte applicazioni, due parti si scambiano una serie di messaggi utilizzando la stessa chiave segreta . È fondamentale garantire che la sicurezza non si degradi anche se un avversario osserva l'intera sequenza di cifrati . Per formalizzare questa nozione, estendiamo l'esperimento di indistinguibilità che abbiamo già visto.

**Definizione** (Esperimento di Indistinguibilità per Messaggi Multipli $\text{PrivK}^{\text{eav-mult}}_{A,\Pi}(n)$)**.** *Per uno schema* $\Pi = (Gen, Enc, Dec)$ *e un avversario* $A$*, l'esperimento è definito come segue :*

1. *L'avversario* $A(1^n)$ *genera due liste di messaggi di uguale lunghezza* $t$*:* $M_0 = (m_{0,1}, \ldots, m_{0,t})$ *e* $M_1 = (m_{1,1}, \ldots, m_{1,t})$*, tali che* $\mid m_{0,i} \mid = \mid m_{1,i} \mid$ *per ogni* $i = 1, \ldots, t$ *.*
2. *Il Challenger genera una chiave* $k \leftarrow Gen(1^n)$ *e sceglie un bit casuale* $b \leftarrow \{0,1\}$*. Calcola quindi una lista di cifrati* $C = (c_1, \ldots, c_t)$ *dove* $c_i \leftarrow Enc_k(m_{b,i})$ *per ogni* $i$ *.*
3. *L'avversario* $A$ *riceve la lista di cifrati* $C$ *e restituisce un bit* $b'$ *.*
4. *L'output dell'esperimento è 1 se* $b' = b$ *(l'avversario ha successo), e 0 altrimenti .*

La sicurezza richiede che nessun avversario efficiente possa vincere questo gioco con una probabilità significativamente maggiore di 1/2 .

**Definizione** (IND-EAV-MULT - Sicurezza per Cifrature Multiple)**.** *Uno schema di cifratura a chiave privata* $\Pi = (Gen, Enc, Dec)$ *ha* ***cifrature multiple indistinguibili in presenza di un eavesdropper*** *se, per ogni avversario* $A$ *PPT, esiste una funzione trascurabile* $negl$ *tale che :*

$$Pr[PrivK^{eav\text{-}mult}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Questa nozione è strettamente più forte della sicurezza per messaggio singolo (IND-EAV), poiché quest'ultima è un caso speciale della prima con $t = 1$ . L'implicazione inversa, tuttavia, non vale, come dimostrato dal seguente controesempio.

**Proposizione.** *Esiste uno schema di cifratura che è IND-EAV sicuro ma non IND-EAV-MULT sicuro .*

**Dimostrazione.** *Consideriamo lo schema One-Time Pad (OTP) . Grazie alla sua segretezza perfetta, l'OTP è anche IND-EAV sicuro . Tuttavia, non è sicuro per cifrature multiple . Consideriamo un avversario A che, nell'esperimento* $PrivK^{eav\text{-}mult}_{A,otp}$*, agisce come segue:*

1. *A sceglie le liste* $M_0 = (0^l, 0^l)$ *e* $M_1 = (0^l, 1^l)$ *.*
2. *A riceve la lista di cifrati* $C = (c_1, c_2)$ *.*
3. *Se* $c_1 = c_2$*, A restituisce* $b' = 0$*; altrimenti, restituisce* $b' = 1$ *.*

*L'OTP è uno schema deterministico:* $c = m \oplus k$*. Analizziamo quindi il successo di A:*

- *Se* $b = 0$*, i messaggi sono entrambi* $0^l$*. Quindi* $c_1 = 0^l \oplus k$ *e* $c_2 = 0^l \oplus k$*, il che implica* $c_1 = c_2$*. A restituisce* $b' = 0$ *e vince .*
- *Se* $b = 1$*, i messaggi sono* $0^l$ *e* $1^l$*. Quindi* $c_1 = 0^l \oplus k$ *e* $c_2 = 1^l \oplus k$*, il che implica* $c_1 \neq c_2$*. A restituisce* $b' = 1$ *e vince .*

*In entrambi i casi, A vince con probabilità 1 . Pertanto, l'OTP non è IND-EAV-MULT sicuro.*

Questo attacco sfrutta la natura deterministica della cifratura . Se lo stesso messaggio viene cifrato due volte con la stessa chiave, si ottiene sempre lo stesso cifrato. Questo porta a una conclusione fondamentale.

**Teorema.** *Se* $\Pi$ *è uno schema di cifratura in cui la funzione Enc è deterministica, allora* $\Pi$ *non può avere cifrature multiple indistinguibili .*

Per ottenere la sicurezza per messaggi multipli, è essenziale che la cifratura sia **probabilistica**: lo stesso messaggio, cifrato più volte con la stessa chiave, deve produrre cifrati diversi .

## 23.2 Attacchi con Testo in Chiaro Scelto (Chosen-Plaintext Attacks - CPA)

Un modello di minaccia più potente e realistico è quello in cui l'avversario può influenzare o scegliere i testi in chiaro che le parti oneste cifrano . Questo scenario è noto come **Chosen-Plaintext Attack (CPA)** .

### 23.2.1 Oracoli: Un Nuovo Strumento di Modellazione

Per modellare la capacità dell'avversario di ottenere i cifrati di messaggi a sua scelta, introduciamo il concetto di **oracolo** . Un oracolo è una "scatola nera" concettuale che esegue un'operazione crittografica per conto dell'avversario. Nel contesto CPA, l'avversario ha accesso a un **oracolo di cifratura**, denotato $\text{Enc}_k(\cdot)$, che, dato un messaggio $m$ in input, restituisce un cifrato $c \leftarrow \text{Enc}_k(m)$ calcolato con una chiave $k$ sconosciuta all'avversario . L'avversario può interrogare (fare una *query*) l'oracolo un numero polinomiale di volte, anche in modo *adattivo*, cioè scegliendo le query successive in base alle risposte precedenti .

### 23.2.2 Definizione di Sicurezza CPA (IND-CPA)

L'esperimento di indistinguibilità viene modificato per concedere all'avversario l'accesso a un oracolo di cifratura .

**Definizione** (Esperimento di Indistinguibilità CPA $\text{PrivK}^{\text{cpa}}_{A,\Pi}(n)$)**.** *Per uno schema* $\Pi = (Gen, Enc, Dec)$ *e un avversario A, l'esperimento è definito come segue :*

1. *Il Challenger genera una chiave* $k \leftarrow Gen(1^n)$ .
2. *All'avversario A viene dato l'input* $1^n$ *e l'accesso all'oracolo* $Enc_k(\cdot)$*. A sceglie due messaggi* $m_0, m_1$ *di uguale lunghezza .*
3. *Il Challenger sceglie un bit casuale* $b \leftarrow \{0,1\}$ *e calcola il cifrato di sfida* $c \leftarrow Enc_k(m_b)$ .
4. *A riceve c e può continuare a interrogare l'oracolo* $Enc_k(\cdot)$*. Alla fine, restituisce un bit* $b'$ .
5. *L'output dell'esperimento è 1 se* $b' = b$*, e 0 altrimenti .*

**Definizione** (IND-CPA - Sicurezza contro Attacchi Chosen-Plaintext)**.** *Uno schema di cifratura a chiave privata* $\Pi$ *è* ***CPA-sicuro*** *se, per ogni avversario A PPT, esiste una funzione trascurabile negl tale che :*

$$Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

La sicurezza CPA è oggi considerata il requisito minimo standard per qualsiasi schema di cifratura a chiave simmetrica, poiché modella un avversario molto potente ma realistico .

## 23.3 Equivalenza tra Sicurezza CPA Singola e Multipla

Analogamente a quanto visto per la sicurezza EAV, possiamo definire un esperimento per la sicurezza CPA per messaggi multipli, $\text{PrivK}^{\text{LR-cpa}}_{A,\Pi}(n)$, dove l'avversario interagisce con

un oracolo "Left-or-Right" $\mathrm{LR}_{k,b}(\cdot,\cdot)$ . Questo oracolo, su input $(m_0, m_1)$, cifra sempre il messaggio "di sinistra" $(m_0)$ se $b = 0$, o sempre quello "di destra" $(m_1)$ se $b = 1$ .

Un risultato fondamentale è che, a differenza del caso EAV, nel contesto CPA la sicurezza per un singolo messaggio di sfida implica la sicurezza per messaggi multipli.

**Teorema** (Equivalenza della Sicurezza CPA)**.** *Uno schema di cifratura a chiave privata è CPA-sicuro se e solo se è CPA-sicuro per cifrature multiple .*

Questo teorema ha importanti conseguenze pratiche:

- Semplifica le dimostrazioni di sicurezza: è sufficiente provare la sicurezza per un singolo messaggio di sfida per garantire la sicurezza anche quando lo schema viene usato per cifrare molti messaggi .
- Ci permette di concentrarci sulla costruzione di schemi per messaggi di lunghezza fissa, sapendo che possono essere estesi in modo sicuro a messaggi di lunghezza arbitraria . Ad esempio, per cifrare un messaggio $m = m_1...m_l$, si può semplicemente concatenare le cifrature dei singoli bit: $\mathrm{Enc}'_k(m) = \mathrm{Enc}_k(m_1) \,||\, \ldots \,||\, \mathrm{Enc}_k(m_l)$ .

# 24 Lezione 8: Costruire Schemi CPA-Sicuri con Funzioni Pseudocasuali

Per costruire uno schema di cifratura che sia CPA-sicuro, abbiamo bisogno di una primitiva crittografica più potente di un generatore pseudocasuale: la **funzione pseudocasuale (PRF)** . Mentre un PRG genera una singola stringa "che sembra casuale", una PRF simula una famiglia di funzioni "che sembrano casuali" .

## 24.1 Funzioni Pseudocasuali (PRF)

Una PRF è una funzione parametrizzata da una chiave che, per una chiave scelta a caso, è computazionalmente indistinguibile da una funzione scelta a caso dall'insieme di tutte le possibili funzioni con lo stesso dominio e codominio .

**Definizione** (Funzione Parametrizzata da Chiave (Keyed Function))**.** *Una funzione parametrizzata da chiave $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ è un algoritmo efficiente che prende in input una chiave $k$ e un input $x$ e restituisce un output $F(k,x)$ . Per una chiave $k$ fissata, $F_k(x) \triangleq F(k,x)$ definisce una funzione a singolo argomento . Per semplicità, consideriamo funzioni che preservano la lunghezza, dove $| k |=| x |=| F_k(x) |= n$ .*

Per definire la pseudocasualità, confrontiamo $F_k$ (per $k$ casuale) con una funzione $f$ scelta uniformemente a caso dall'insieme $\mathrm{Func}_n$ di tutte le funzioni da $\{0,1\}^n$ a $\{0,1\}^n$ . La cardinalità di $\mathrm{Func}_n$ è $2^{n \cdot 2^n}$, un numero enorme, rendendo impossibile dare la sua descrizione completa a un distinguisher . La soluzione è, ancora una volta, usare un oracolo .

**Definizione** (Funzione Pseudocasuale (PRF)). *Una funzione con chiave efficiente e che preserva la lunghezza $F$ è una* ***funzione pseudocasuale (PRF)*** *se per ogni distinguisher $D$ PPT, esiste una funzione trascurabile $negl$ tale che:*

$$\mid Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1] \mid \leq negl(n)$$

*dove la prima probabilità è calcolata sulla scelta uniforme di $k \in \{0,1\}^n$, e la seconda sulla scelta uniforme di $f \in Func_n$ .*

**Esempio** (Funzione Non-PRF). *La funzione $F_k(x) = k \oplus x$ non è una PRF . Sebbene per un singolo $x$ l'output sia uniforme, esiste una correlazione tra gli output per input diversi. Un distinguisher $D$ può fare due query, $x_1$ e $x_2 \neq x_1$, ottenendo $y_1 = O(x_1)$ e $y_2 = O(x_2)$. Se $y_1 \oplus y_2 = x_1 \oplus x_2$, $D$ indovina che l'oracolo implementa $F_k$ .*

- *Se $O(\cdot) = F_k(\cdot)$, allora $y_1 \oplus y_2 = (k \oplus x_1) \oplus (k \oplus x_2) = x_1 \oplus x_2$ è sempre vero. Quindi $D$ restituisce 1 con probabilità 1 .*
- *Se $O(\cdot) = f(\cdot)$, allora $y_1$ e $y_2$ sono valori casuali e indipendenti. La probabilità che $y_1 \oplus y_2 = x_1 \oplus x_2$ è esattamente $1/2^n$ .*

*La differenza nelle probabilità di successo, $\mid 1 - 1/2^n \mid$, non è trascurabile .*

### 24.1.1 Permutazioni Pseudocasuali (PRP)

Una **permutazione pseudocasuale (PRP)** è una PRF $F$ tale che per ogni chiave $k$, la funzione $F_k$ è una permutazione (una biezione) . Una **PRP forte (SPRP)** è una PRP che rimane indistinguibile da una permutazione casuale anche se il distinguisher ha accesso sia all'oracolo diretto ($F_k(\cdot)$) sia a quello inverso ($F_k^{-1}(\cdot)$) . I cifrari a blocchi (block ciphers) usati in pratica sono modellati come SPRP .

## 24.2 Costruzione di uno Schema CPA-Sicuro da una PRF

Avendo a disposizione una PRF, possiamo costruire uno schema di cifratura CPA-sicuro. Il tentativo più semplice, $Enc_k(m) = F_k(m)$, fallisce perché è deterministico . L'approccio corretto consiste nell'introdurre casualità nella cifratura, usando la PRF per generare un "pad" pseudocasuale diverso per ogni operazione di cifratura .

**Definizione** (Costruzione CPA-sicura da PRF (Costruzione 3.28 in )). *Sia $F$ una PRF. Definiamo uno schema di cifratura per messaggi di lunghezza $n$ come segue :*

- ***Gen($1^n$)****: Sceglie una chiave uniforme $k \in \{0,1\}^n$ .*
- ***Enc$_k$(m)****: Sceglie una stringa uniforme $r \in \{0,1\}^n$ e restituisce il cifrato $c := \langle r, F_k(r) \oplus m \rangle$ .*
- ***Dec$_k$(c)****: Interpreta il cifrato $c$ come $\langle r, s \rangle$ e restituisce il messaggio $m := F_k(r) \oplus s$ .*

Questo schema è corretto perché $(F_k(r) \oplus m) \oplus F_k(r) = m$. Poiché a ogni cifratura viene scelta una nuova stringa casuale $r$, lo schema è probabilistico e supera il limite dei cifrari deterministici .

**Teorema.** *Se $F$ è una funzione pseudocasuale, la costruzione sopra è uno schema di cifratura CPA-sicuro .*

**Dimostrazione** (Dimostrazione (schizzo))**.** *La prova si basa su una riduzione e segue una strategia in due fasi .*

***Fase 1: Sostituzione con una funzione casuale.*** *Si definisce uno schema "ipotetico" $\tilde{\Pi}$ identico a quello dato, ma dove la PRF $F_k$ è sostituita da una funzione $f$ scelta uniformemente a caso da $Func_n$ . Si dimostra per riduzione che se un avversario $A$ può distinguere tra una cifratura di $m_0$ e una di $m_1$ nello schema reale $\Pi$ con probabilità non trascurabile, allora si può costruire un distinguisher $D$ che distingue tra $F_k$ e $f$ . Il distinguisher $D$ simula l'esperimento CPA per $A$ . Quando $A$ fa una query (sia all'oracolo di cifratura che per la sfida), $D$ usa il proprio oracolo $O(\cdot)$ (che è $F_k$ o $f$) per generare il pad e costruire il cifrato . Poiché per ipotesi $F$ è una PRF, la differenza di comportamento di $A$ nei due scenari (reale e ipotetico) deve essere trascurabile .*

$$\mid Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] - Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] \mid \leq negl(n)$$

***Fase 2: Analisi dello schema ipotetico.*** *Si analizza la sicurezza dello schema ipotetico $\tilde{\Pi}$ che usa una funzione veramente casuale $f$ . Il cifrato di sfida è $c^* = \langle r^*, f(r^*) \oplus m_b \rangle$. La sicurezza si basa sul fatto che, con probabilità molto alta, il valore casuale $r^*$ scelto per la sfida non è mai stato usato per nessuna delle query all'oracolo di cifratura fatte da $A$ . Sia $q(n)$ il numero di query all'oracolo. La probabilità che $r^*$ coincida con uno degli $r$ usati per le query è al più $q(n)/2^n$ .*

- *Se $r^*$ non è mai stato usato prima (evento che accade con probabilità $\geq 1 - q(n)/2^n$), allora $f(r^*)$ è un valore uniformemente casuale e indipendente da tutto ciò che l'avversario ha visto . Il pad $f(r^*)$ maschera perfettamente $m_b$, come in un One-Time Pad, e la probabilità che $A$ indovini $b$ è esattamente 1/2 .*
- *Se $r^*$ è stato usato in una query precedente (evento Repeat), l'avversario può conoscere $f(r^*)$ e vincere con probabilità 1 . Questo evento, però, accade con probabilità al più $q(n)/2^n$ .*

*La probabilità totale di successo di $A$ contro lo schema ipotetico è quindi limitata superiormente da:*

$$Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

*Poiché $q(n)$ è polinomiale, $q(n)/2^n$ è una funzione trascurabile.*

***Conclusione.*** *Unendo i risultati delle due fasi, la probabilità di successo di $A$ contro lo schema reale $\Pi$ è:*

$$Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] + negl_1(n) \leq \frac{1}{2} + \frac{q(n)}{2^n} + negl_1(n)$$

*La somma di funzioni trascurabili è ancora trascurabile, quindi lo schema è CPA-sicuro .*

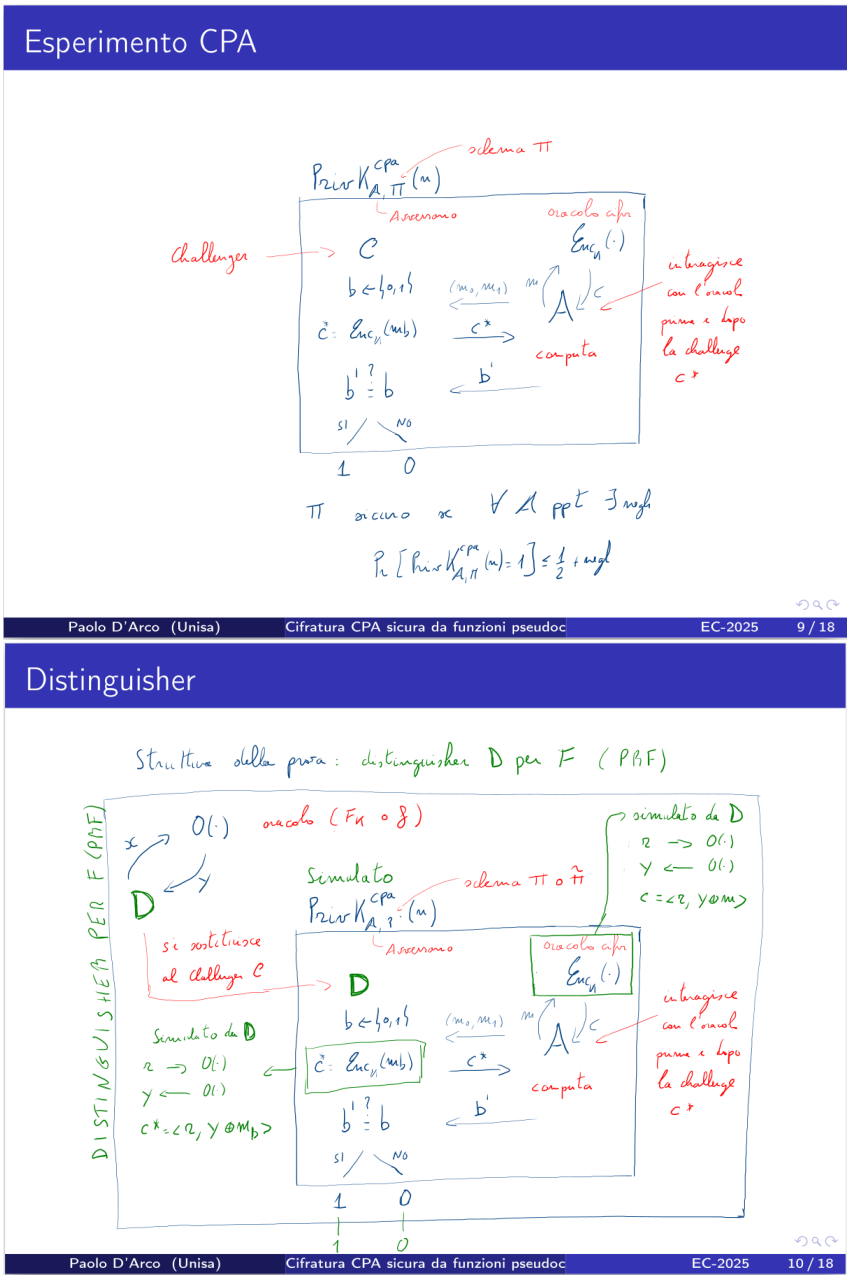


Figura 1: Schema di cifratura CPA-sicuro basato su PRF .

# 25 Lezione 9: Modalità operative di cifratura

Nelle lezioni precedenti, abbiamo definito e costruito schemi di cifratura computazionalmente sicuri (EAV e CPA) per messaggi di lunghezza fissa. Ad esempio, la Costruzione 3.28, basata su una PRF, è CPA-sicura ma presenta uno svantaggio: raddoppia la lunghezza del messaggio nel cifrato ($| c | = 2 | m |$). Se volessimo utilizzare tale schema per cifrare un messaggio lungo, potremmo suddividerlo in blocchi e cifrare ciascun blocco individualmente, ma ciò comporterebbe un'inefficiente espansione del cifrato.

Le **modalità operative di cifratura** sono metodi standardizzati e efficienti che per-

mettono di utilizzare primitive crittografiche a lunghezza fissa, come gli *stream cipher* e i *cifrari a blocchi*, per cifrare in modo sicuro messaggi di lunghezza arbitraria, minimizzando l'espansione del testo cifrato.

## 25.1 Modalità operative per Stream Cipher

Uno stream cipher (o un PRG) viene utilizzato per generare un "pad" pseudocasuale da porre in XOR con il messaggio, replicando in ambito computazionale il concetto del One-Time Pad. Per gestire la cifratura di messaggi multipli, esistono due approcci principali.

### 25.1.1 Modalità Sincrona (con stato)

In questa modalità, le due parti comunicanti generano una lunga sequenza pseudocasuale a partire da una chiave segreta condivisa $k$ e la utilizzano in porzioni successive per cifrare diversi messaggi.

- **Funzionamento:** Le parti inizializzano lo stream cipher con la chiave $k$, ottenendo uno stato iniziale $st_0 := \text{Init}(k)$. Per cifrare il primo messaggio $m_1$ di lunghezza $l_1$, il mittente invoca $\text{GetBits}(st_0, 1^{l_1})$ per generare un $pad_1$ e calcola $c_1 := pad_1 \oplus m_1$. Lo stato interno dello stream cipher viene quindi aggiornato a $st_1$. Per cifrare un secondo messaggio $m_2$, il processo riparte dallo stato corrente $st_1$ per generare un nuovo pad, e così via per i messaggi successivi.
- **Requisiti:** È fondamentale che entrambe le parti mantengano e aggiornino lo stato dello stream cipher in modo perfettamente sincronizzato.
- **Sicurezza:** Se lo stream cipher sottostante è un PRG sicuro, lo schema di cifratura risultante è **EAV-sicuro**.
- **Utilizzo:** Questa modalità è efficiente e adatta per sessioni di comunicazione continue tra due parti. Risulta però problematica per comunicazioni sporadiche o quando una parte potrebbe utilizzare dispositivi fisici diversi, in quanto diventa difficile garantire la sincronia dello stato.

### 25.1.2 Modalità Asincrona (senza stato)

Per superare i limiti della modalità sincrona, la modalità asincrona non richiede di mantenere uno stato tra una cifratura e l'altra. Al suo posto, per ogni nuovo messaggio, viene scelto un **Vettore di Inizializzazione (IV)** unico e casuale.

- **Funzionamento:** Per cifrare un messaggio $m$ con chiave $k$, il mittente sceglie un IV casuale, che non deve mai essere riutilizzato per la stessa chiave. Lo stream cipher viene inizializzato con entrambi: $st_0 := \text{Init}(k, \text{IV})$. Viene generato il *pad* della lunghezza necessaria e si calcola $c' = pad \oplus m$. Il cifrato finale trasmesso è

la coppia $c = \langle \text{IV}, c' \rangle$. L'IV viene inviato in chiaro insieme al messaggio cifrato, in modo che il destinatario possa utilizzarlo per generare lo stesso pad e decifrare correttamente.

- **Sicurezza:** Se la funzione $F_k(\text{IV}) \stackrel{\text{def}}{=} G_\infty(k, \text{IV}, 1^{|m|})$ si comporta come una **funzione pseudocasuale (PRF)**, allora lo schema di cifratura risultante è **CPA-sicuro**.
- **Osservazione (PRF Debole):** Poiché l'IV viene scelto uniformemente a caso e non secondo una distribuzione scelta dall'avversario, è sufficiente che $F_k$ sia una **PRF debole** (*weakly pseudorandom*), ovvero una funzione che risulta pseudocasuale solo quando i suoi input sono scelti uniformemente a caso.

## 25.2 Modalità operative per Cifrari a Blocchi

Le modalità operative per i cifrari a blocchi, come AES, sono metodi standard per cifrare messaggi di lunghezza arbitraria. Se un messaggio non è un multiplo intero della lunghezza del blocco $n$, deve essere prima esteso tramite una tecnica di **padding** (es. aggiungendo un '1' seguito da tanti '0' quanti necessari).

### 25.2.1 Electronic Code Book (ECB) Mode

È la modalità più semplice: il messaggio $m = m_1 \,||\, \ldots \,||\, m_l$ viene diviso in blocchi e il cifrario a blocchi $F_k$ è applicato a ciascun blocco in modo indipendente.

$$c_i := F_k(m_i)$$

**Analisi:** La modalità ECB è **deterministica**, il che rappresenta una vulnerabilità critica. Se un blocco di testo in chiaro si ripete ($m_i = m_j$), anche il corrispondente blocco di testo cifrato si ripeterà ($c_i = c_j$). Questo fa trapelare importanti informazioni sulla struttura del messaggio, rendendo lo schema non solo non CPA-sicuro, ma persino non EAV-sicuro. Un avversario può facilmente vincere il gioco dell'indistinguibilità costruendo due messaggi, uno con blocchi tutti distinti e uno con almeno un blocco ripetuto. **La modalità ECB non dovrebbe mai essere utilizzata**.

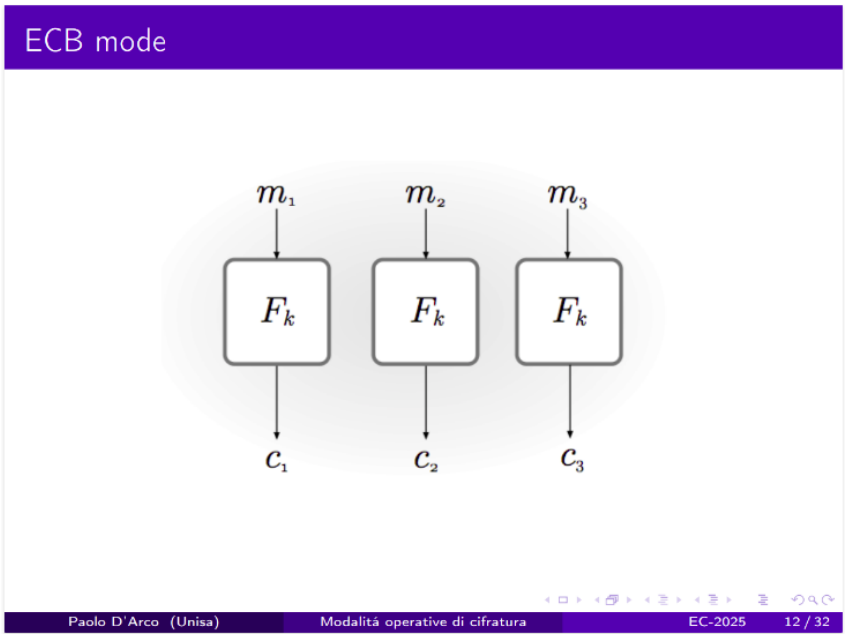


Figura 2: Modalità Electronic Code Book (ECB).

### 25.2.2 Cipher Block Chaining (CBC) Mode

La modalità CBC (Cipher Block Chaining) risolve il problema del determinismo dell'ECB introducendo una dipendenza tra la cifratura dei blocchi.

- **Cifratura:** Viene scelto un IV casuale e uniforme di lunghezza $n$. Il primo blocco $m_1$ è posto in XOR con l'IV prima di essere cifrato. Per i blocchi successivi, ogni blocco $m_i$ è posto in XOR con il blocco cifrato *precedente* $c_{i-1}$ prima della cifratura.

$$c_0 := \text{IV}, \quad c_i := F_k(c_{i-1} \oplus m_i) \quad \text{per } i = 1, \ldots, l$$

  Il cifrato finale è $\langle \text{IV}, c_1, \ldots, c_l \rangle$.

- **Decifratura:** Richiede che $F$ sia una permutazione, poiché è necessario calcolare l'inversa $F_k^{-1}$:

$$m_i := F_k^{-1}(c_i) \oplus c_{i-1}$$

- **Analisi:** Grazie all'IV casuale, la cifratura CBC è **probabilistica**. Se $F$ è una permutazione pseudocasuale (PRP), la modalità CBC è **CPA-sicura**. Lo svantaggio principale è che la cifratura è **intrinsecamente sequenziale**, non parallelizzabile, poiché il calcolo di $c_i$ dipende da $c_{i-1}$. La decifratura, invece, può essere parallelizzata.

**Attenzione:** Una variante chiamata *Chained CBC*, in cui l'ultimo blocco del cifrato precedente viene usato come IV per il messaggio successivo, **non è CPA-sicura** ed è vulnerabile ad attacchi a testo in chiaro scelto, come dimostrato in SSL 3.0 e TLS 1.0.

### 25.2.3 Output Feedback (OFB) Mode

La modalità OFB trasforma un cifrario a blocchi in uno **stream cipher sincrono**.

- **Funzionamento:** A partire da un IV, si genera una sequenza di blocchi di "pad" pseudocasuale ($y_i$) applicando iterativamente il cifrario a blocchi. Il messaggio viene poi cifrato tramite XOR con questo pad.

$$y_0 := \text{IV}, \quad y_i := F_k(y_{i-1}), \quad c_i := m_i \oplus y_i$$

- **Analisi:** La decifratura è identica alla cifratura e $F$ **non deve essere invertibile**. Il pad pseudocasuale può essere **pre-calcolato** indipendentemente dal messaggio. Se $F$ è una PRF, la modalità OFB è **CPA-sicura**.

### 25.2.4 Counter (CTR) Mode

La modalità contatore (CTR) è una delle più moderne, efficienti e raccomandate. Trasforma un cifrario a blocchi in uno **stream cipher asincrono** ad alte prestazioni e completamente parallelizzabile.

- **Funzionamento:** Si sceglie un valore iniziale casuale per un contatore ('ctr'), che funge da nonce. Per ogni blocco $i$, il pad viene generato cifrando un valore unico, tipicamente 'ctr + i'.

$$y_i := F_k(\text{ctr} + i \pmod{2^n}), \quad c_i := m_i \oplus y_i$$

  Il cifrato finale è $\langle \text{ctr}, c_1, \ldots, c_l \rangle$.

- **Analisi:**
  - È **altamente parallelizzabile**: ogni blocco di pad $y_i$ può essere calcolato indipendentemente dagli altri, sia in cifratura che in decifratura. Questo permette anche l'**accesso casuale** ai blocchi del messaggio.
  - $F$ non necessita di essere invertibile.
  - Il pad può essere pre-calcolato.

**Teorema** (3.32, Katz-Lindell)**.** *Se $F$ è una funzione pseudocasuale, allora la modalità CTR è CPA-sicura.*

**Dimostrazione** (Dimostrazione (schizzo))**.** *La dimostrazione si basa sulla stessa strategia usata in precedenza, con un argomento ibrido.*

1. ***Schema Ipotetico*** $\tilde{\Pi}$***:*** *Si definisce uno schema identico alla modalità CTR, ma dove la PRF $F_k$ è sostituita da una funzione $f$ scelta uniformemente a caso da $Func_n$.*

2. ***Riduzione a PRF:*** *Si dimostra che se un avversario A potesse distinguere tra una cifratura in* $\Pi$ *(con* $F_k$*) e una in* $\tilde{\Pi}$ *(con* $f$*) con vantaggio non trascurabile, allora A potrebbe essere usato per costruire un distinguisher D che rompe la sicurezza della PRF F. Formalmente:*

$$| Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] - Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] | \leq negl(n)$$

3. ***Analisi di*** $\tilde{\Pi}$***:*** *Si analizza la sicurezza dello schema ipotetico. Poiché* $f$ *è una funzione veramente casuale e il valore del contatore* $(ctr + i)$ *è unico per ogni blocco (con probabilità molto alta), ogni blocco di pad* $y_i = f(ctr + i)$ *è un valore uniforme e indipendente da tutti gli altri. Il messaggio è quindi mascherato con un pad che è indistinguibile da un One-Time Pad. La probabilità che un avversario vinca è limitata dalla probabilità di una collisione sui valori del contatore, che è trascurabile.*

$$Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] \leq \frac{1}{2} + \frac{2 \cdot q(n)^2}{2^n}$$

*dove* $q(n)$ *è un limite superiore al numero totale di blocchi cifrati dall'avversario.*

4. ***Conclusione:*** *Combinando i due passi, si conclude che la probabilità di successo dell'avversario contro la modalità CTR reale è al più* $1/2$ *più una funzione trascurabile.*

## 25.3 Cifratura Nonce-Based

Uno schema di cifratura *nonce-based* (number used once) adotta una sintassi alternativa in cui l'algoritmo di cifratura è **deterministico**, ma accetta un input aggiuntivo, il `nonce`, oltre alla chiave e al messaggio.

$$c := \text{Enc}_k(\text{nonce}, m)$$

La condizione di sicurezza fondamentale è che **il nonce non deve mai essere ripetuto per la stessa chiave**. L'onere di garantire questa unicità è trasferito all'applicazione che utilizza lo schema crittografico.

La modalità CTR è un esempio perfetto di schema nonce-based CPA-sicuro, dove il valore iniziale del contatore 'ctr' funge da 'nonce'. Il fatto che uno schema deterministico come questo possa essere CPA-sicuro non contraddice il Teorema 3.20, perché la casualità richiesta è introdotta dal nonce unico, anziché da un algoritmo di cifratura probabilistico.

**Vantaggi degli schemi nonce-based:**

- Sono utili in contesti in cui generare casualità di alta qualità è difficile o costoso (es. dispositivi embedded), mentre mantenere un contatore è semplice.
- Per applicazioni che necessitano di cifrare pochi messaggi, è sufficiente un nonce di piccole dimensioni.
- Permettono di ottenere riduzioni di sicurezza più "strette" (*tighter*), ovvero più efficienti nell'analisi della sicurezza concreta.

### 25.4 Esercizi sulla Lezione 9

1. Sia $G : \{0,1\}^n \to \{0,1\}^l$, con $l > n$, un PRG e sia $G_0 : \{0,1\}^{2n} \to \{0,1\}^{n+l}$ definito come $G_0(r \mid\mid s) = r \mid\mid G(s)$, con $r, s \in \{0,1\}^n$. È $G_0$ un PRG?
2. Relativamente all'Esempio 3.6 del libro di testo (un generatore non sicuro), ideare un altro modo per costruire un distinguisher D efficiente ed altrettanto efficace.
3. Sia $F(k,x) = k \oplus x^c$, per qualche costante $c > 1$. È una PRF? E se si sostituisce $x^c$ con un certo polinomio $p(x)$?
4. Provare da soli la prima parte della dimostrazione del Teorema 3.32 (sicurezza della modalità CTR), rivedendo i passi della dimostrazione del Teorema 3.31 (costruzione di cifrario da PRG), se necessario.

## 26 Introduzione: Dalla Confidenzialità all'Integrità

Nelle lezioni precedenti, abbiamo esplorato come la crittografia a chiave simmetrica possa garantire la **confidenzialità** dei dati, proteggendo il contenuto dei messaggi da un avversario passivo (eavesdropper). Abbiamo raggiunto questo obiettivo attraverso schemi di cifratura che soddisfano la nozione di sicurezza **CPA (Chosen-Plaintext Attack)**, come quelli basati su funzioni pseudocasuali (PRF) o implementati tramite modalità operative quali CTR e CBC.

Tuttavia, una comunicazione può essere definita "sicura" solo se, oltre alla confidenzialità, garantisce anche altre due proprietà fondamentali: **autenticità** e **integrità**.

- **Autenticità**: Garantisce che il mittente del messaggio sia effettivamente chi dichiara di essere.
- **Integrità**: Assicura che il messaggio ricevuto sia identico a quello inviato, senza alterazioni, inserzioni o cancellazioni.

Queste proprietà sono cruciali in scenari reali. Ad esempio, quando un utente invia una richiesta di bonifico alla propria banca, è essenziale che la banca possa verificare sia l'identità dell'utente sia che l'importo non sia stato alterato durante la trasmissione. Similmente, un server di e-commerce deve potersi fidare dell'integrità dei cookie memorizzati sul client, per evitare che un utente malintenzionato possa, ad esempio, modificare il prezzo di un articolo nel proprio carrello.

### 26.1 L'Insufficienza della Cifratura per l'Integrità

È un errore comune pensare che la cifratura, da sola, possa garantire l'integrità. L'intuizione errata è che se un avversario non può leggere il messaggio, non può nemmeno modificarlo in modo sensato. Tutti gli schemi di cifratura che abbiamo studiato finora sono vulnerabili a manipolazioni che compromettono l'integrità.

**Schemi basati su XOR (Stream Cipher, OFB, CTR):** In schemi dove il cifrato è $c = \text{pad} \oplus m$, un avversario può invertire un bit $i$ del messaggio in chiaro semplicemente invertendo il bit $i$ del cifrato, senza conoscere né la chiave né il messaggio originale. Questa malleabilità può avere conseguenze disastrose, come alterare l'importo di una transazione finanziaria.

**Modalità CBC:** Anche in modalità CBC, sebbene le manipolazioni siano più complesse, un avversario può ottenere effetti controllati. Ad esempio, modificando il j-esimo bit del vettore di inizializzazione (IV), si ottiene la modifica controllata del j-esimo bit del primo blocco di messaggio $m_1$, poiché $m_1 = F_k^{-1}(c_1) \oplus IV$.

Inoltre, tutti gli schemi visti finora sono tali che quasi ogni stringa di bit della lunghezza appropriata è un cifrato valido per qualche messaggio. Un avversario può quindi "inventare" un cifrato, inviarlo, e il destinatario lo decifrerà ottenendo un messaggio (probabilmente senza senso), senza però accorgersi della falsificazione. Per garantire l'integrità, è necessario uno strumento crittografico specifico.

# 27 Codici per l'Autenticazione di Messaggi (MAC)

Un **Message Authentication Code (MAC)** è una primitiva crittografica progettata per fornire autenticità e integrità in un contesto a chiave simmetrica. Le due parti condividono una chiave segreta $k$. Il mittente, Alice, calcola un *tag* $t$ sul messaggio $m$ usando la chiave $k$ e lo invia insieme al messaggio. Il destinatario, Bob, usa la stessa chiave $k$ per verificare che il tag $t$ sia valido per il messaggio $m$.

## 27.1 Sintassi e Definizione Formale

**Definizione** (Schema MAC - Definizione 4.1 in Katz/Lindell)**.** *Uno schema per l'autenticazione di messaggi (MAC) consiste in tre algoritmi a tempo polinomiale probabilistico (PPT):* $\Pi = (Gen, Mac, Vrfy)$.

1. ***Generazione della Chiave (Gen):*** $k \leftarrow Gen(1^n)$. *È un algoritmo probabilistico che, dato il parametro di sicurezza* $n$*, genera una chiave* $k$.
2. ***Generazione del Tag (Mac):*** $t \leftarrow Mac_k(m)$. *È un algoritmo (potenzialmente probabilistico) che, data la chiave* $k$ *e un messaggio* $m \in \{0,1\}^*$*, produce un tag* $t$.
3. ***Verifica (Vrfy):*** $b := Vrfy_k(m, t)$. *È un algoritmo deterministico che, dati* $k, m, t$*, restituisce un bit* $b \in \{0,1\}$*, dove* 1 *significa "valido" e* 0 *"non valido".*

*È richiesta la* ***correttezza****: per ogni* $n$*, per ogni chiave* $k$ *generata da* $Gen(1^n)$ *e per ogni messaggio* $m$*, deve valere* $Vrfy_k(m, Mac_k(m)) = 1$*. Se lo schema funziona solo per messaggi di lunghezza fissa* $l(n)$*, si parla di* ***MAC a lunghezza fissa****.*

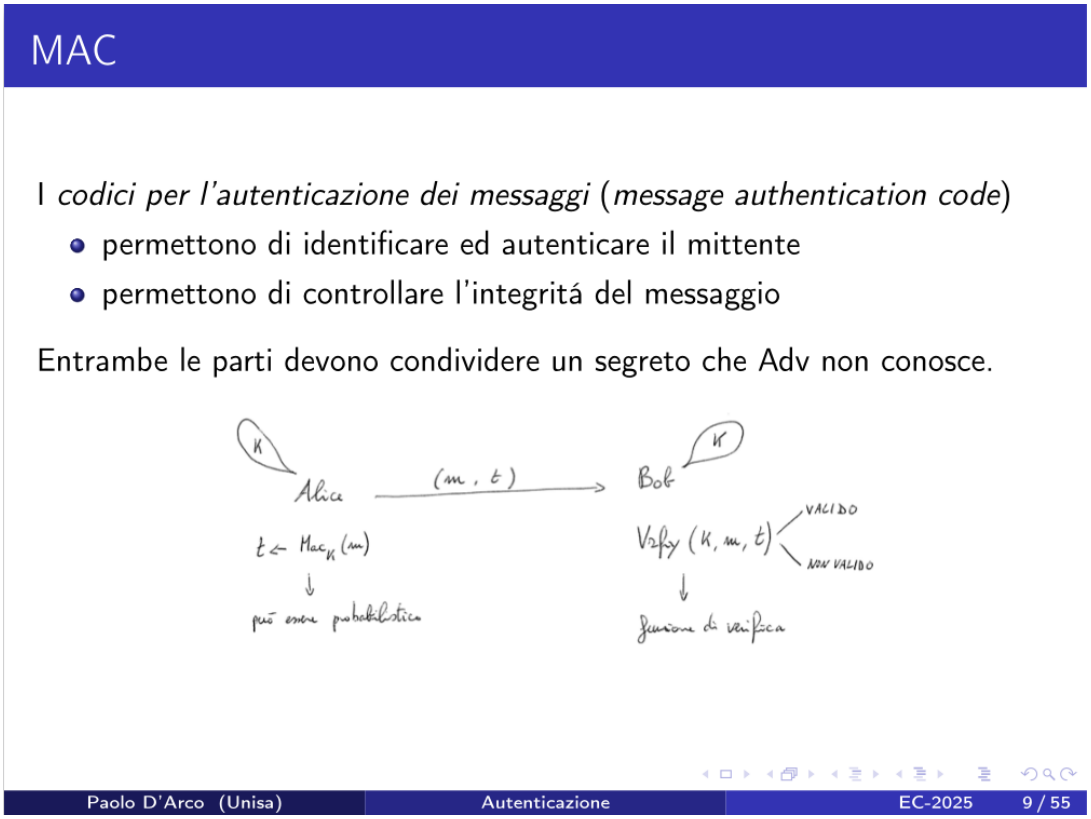


Figura 3: Flusso di un Message Authentication Code.

In molti schemi, l'algoritmo $\text{Mac}_k$ è deterministico. In tal caso, la verifica può essere **canonica**: il verificatore ricalcola il tag $t' := \text{Mac}_k(m)$ e controlla se $t' = t$.

## 27.2 Definizione di Sicurezza: Non Falsificabilità Esistenziale

L'obiettivo di sicurezza per un MAC è impedire a un avversario di creare una *falsificazione* (forgery), ovvero una coppia '(messaggio, tag)' valida per un messaggio che non è stato precedentemente autenticato dal mittente legittimo.

**Potere dell'Avversario:** Si modella un avversario potente che può eseguire un **attacco con messaggi scelti** (chosen-message attack). All'avversario viene dato accesso a un oracolo $\text{Mac}_k(\cdot)$ che, su richiesta di un messaggio $m$, restituisce il tag corrispondente $t$.

**Definizione** (Esperimento di Falsificazione $\text{Mac-forge}_{A,\Pi}(n)$)**.** *Per uno schema MAC* $\Pi = (Gen, Mac, Vrfy)$ *e un avversario* $A$*, l'esperimento è definito come segue:*

1. *Viene generata una chiave* $k \leftarrow Gen(1^n)$.
2. *All'avversario* $A$ *viene dato accesso all'oracolo* $Mac_k(\cdot)$*.* $A$ *può fare un numero polinomiale di query. Sia* $Q$ *l'insieme dei messaggi che* $A$ *ha inviato all'oracolo.*

3. *Alla fine, A produce una coppia* $(m, t)$.

4. *A ha successo (l'output dell'esperimento è 1) se e solo se:*

   - $Vrfy_k(m, t) = 1$ *(il tag è valido).*
   - $m \notin Q$ *(il messaggio è nuovo).*

**Definizione** (Sicurezza di un MAC - Definizione 4.2)**.** *Uno schema MAC* $\Pi$ *è* ***non falsificabile esistenzialmente sotto un attacco adattivo con messaggi scelti*** *(existentially unforgeable under an adaptive chosen-message attack) se, per ogni avversario A PPT, esiste una funzione trascurabile* $negl(n)$ *tale che:*

$$Pr[Mac\text{-}forge_{A,\Pi}(n) = 1] \leq negl(n)$$

**Nota.** *La terminologia "esistenzialmente non falsificabile" significa che l'avversario non deve essere in grado di produrre una falsificazione per nessun nuovo messaggio, anche se questo non ha un significato apparente ("spazzatura"). Questo rende la definizione molto forte e adatta a qualsiasi applicazione.*

### 27.2.1 Attacchi di Replay e Sicurezza Forte

La definizione standard non protegge dagli **attacchi di replay**, in cui un avversario intercetta una coppia $(m, t)$ valida e la reinvia in un secondo momento. La protezione contro i replay deve essere gestita a livello applicativo, ad esempio tramite numeri di sequenza o timestamp.

**Sicurezza Forte (Strong Security):** La definizione standard non impedisce a un avversario di creare un *nuovo tag valido* per un messaggio già autenticato. Una nozione più forte, chiamata **sicurezza forte**, richiede che l'avversario non possa produrre nessuna coppia $(m, t)$ valida che non abbia già ricevuto dall'oracolo. Questo è rilevante solo per i MAC probabilistici.

**Proposizione** (Proposizione 4.4)**.** *Se* $\Pi$ *è un MAC sicuro che usa la verifica canonica, allora* $\Pi$ *è anche fortemente sicuro.*

# 28 Attacchi Side-Channel: Il Timing Attack

Le definizioni formali di sicurezza assumono che l'avversario osservi solo gli input e gli output degli algoritmi. Gli **attacchi side-channel** sfruttano invece informazioni "trapelate" dall'implementazione fisica del sistema, come il tempo di esecuzione, il consumo energetico o la radiazione elettromagnetica.

Un esempio potente è il **timing attack**, che sfrutta le differenze nei tempi di risposta del sistema.

## 28.1 Un Attacco di Temporizzazione su un MAC Canonico

Consideriamo un MAC deterministico con verifica canonica, dove il confronto tra il tag ricevuto $t$ e quello ricalcolato $t'$ è implementato con una funzione simile a 'strcmp' del C. Tale funzione confronta i byte sequenzialmente e si arresta non appena trova una discrepanza.

Un avversario può sfruttare questa caratteristica per ricostruire un tag valido per qualsiasi messaggio $m$, byte per byte.

**Fasi dell'attacco:** Supponiamo che il tag sia lungo 16 byte. L'obiettivo è trovare il tag corretto $t = t_1...t_{16}$ per un messaggio $m$.

1. **Trovare il primo byte ($t_1$):** L'avversario invia 256 coppie $(m, \sigma_j)$ al verificatore, dove $\sigma_j = j \mid\mid$ padding per $j = 0, ..., 255$. Per una sola di queste, $\sigma_{t_1}$, il confronto 'strcmp$(t', \sigma_{t_1})$' durerà leggermente di più, perché fallirà solo al secondo byte. Misurando i tempi di risposta, l'avversario identifica $t_1$.
2. **Trovare i byte successivi:** Una volta noti i primi $i$ byte $t_1...t_i$, l'avversario invia 256 coppie $(m, \sigma_j)$, dove $\sigma_j = t_1...t_i \mid\mid j \mid\mid$ padding. Anche in questo caso, il tentativo che richiederà più tempo per essere rifiutato rivelerà il valore corretto del byte $i + 1$.

Questo attacco richiede al più $16 \times 256 = 4096$ query di verifica per ricostruire un intero tag di 16 byte, rendendolo estremamente pratico. Un attacco simile è stato usato con successo per rompere la protezione dell'Xbox 360.

**Contromisura:** La soluzione è implementare la verifica del tag usando una funzione di confronto a tempo costante, che esamini sempre tutti i byte prima di restituire un risultato, indipendentemente dalla posizione della prima discrepanza.

# 29 Costruzioni di MAC Sicuri

## 29.1 MAC a Lunghezza Fissa da una PRF

Una funzione pseudocasuale (PRF) è uno strumento naturale per costruire un MAC. L'idea è semplice: il tag di un messaggio è semplicemente l'output della PRF calcolata sul messaggio.

***CONSTRUCTION 4.5***

Let $F$ be a pseudorandom function. Define a fixed-length MAC for messages of length $n$ as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$. (If $|m| \neq |k|$ then output nothing.)
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^n$, and a tag $t \in \{0,1\}^n$, output 1 if and only if $t \stackrel{?}{=} F_k(m)$. (If $|m| \neq |k|$, then output 0.)

Figura 4: Costruzione di un MAC a lunghezza fissa da una PRF (Costruzione 4.5).

**Teorema** (Teorema 4.6)**.** *Se $F$ è una funzione pseudocasuale, allora la Costruzione 4.5 è un MAC sicuro a lunghezza fissa per messaggi di lunghezza $n$.*

**Dimostrazione** (Dimostrazione (per riduzione))**.** *Assumiamo per assurdo che la costruzione $\Pi$ non sia sicura. Allora esiste un avversario $A$ che vince l'esperimento Mac-forge$_{A,\Pi}(n)$ con probabilità non trascurabile $\epsilon(n)$. Costruiamo un* ***distinguisher*** *$D$ che usa $A$ per rompere la sicurezza della PRF $F$. $D$ riceve un oracolo $O(\cdot)$ che è o $F_k(\cdot)$ (per un $k$ casuale) o una funzione $f$ scelta a caso da Func$_n$.*

***Algoritmo del Distinguisher*** *$D^{O(\cdot)}(1^n)$**:***

1. *Esegue $A(1^n)$. Ogni volta che $A$ chiede un tag per un messaggio $m_i$, $D$ interroga il suo oracolo $O(m_i)$ e inoltra la risposta a $A$.*
2. *Quando $A$ termina e produce una falsificazione $(m, t)$, $D$ interroga $O(m)$ per ottenere $t' = O(m)$.*
3. *Se $t' = t$ e $m$ non era tra i messaggi chiesti da $A$ all'oracolo, $D$ restituisce 1 (indovinando che $O = F_k$). Altrimenti, $D$ restituisce 0.*

*Analizziamo il comportamento di $D$:*

- ***Se*** *$O(\cdot) = F_k(\cdot)$* ***(oracolo pseudocasuale):*** *La vista di $A$ è identica a quella dell'esperimento Mac-forge$_{A,\Pi}(n)$. Quindi, la probabilità che $D$ restituisca 1 è esattamente la probabilità di successo di $A$:*

$$Pr[D^{F_k(\cdot)}(1^n) = 1] = Pr[\text{Mac-forge}_{A,\Pi}(n) = 1] = \epsilon(n)$$

- ***Se*** *$O(\cdot) = f(\cdot)$* ***(oracolo casuale):*** *La vista di $A$ è quella di un attacco a uno schema MAC ipotetico $\tilde{\Pi}$ basato su una funzione veramente casuale. Per qualsiasi nuovo messaggio $m$, il valore $f(m)$ è uniforme e indipendente da tutto ciò che $A$ ha visto. La probabilità che $A$ indovini il tag corretto è $1/2^n$. Quindi:*

$$Pr[D^{f(\cdot)}(1^n) = 1] = Pr[\text{Mac-forge}_{A,\tilde{\Pi}}(n) = 1] = 1/2^n$$

*Il vantaggio del distinguisher $D$ è:*

$$| \, Pr[D^{F_k(\cdot)} = 1] - Pr[D^{f(\cdot)} = 1] \, | = | \, \epsilon(n) - 1/2^n \, |$$

*Se $\epsilon(n)$ non fosse trascurabile, questo vantaggio non sarebbe trascurabile, contraddicendo l'ipotesi che $F$ sia una PRF. Quindi, $\epsilon(n)$ deve essere trascurabile e la costruzione è sicura.*

## 29.2 Estensione del Dominio: MAC per Messaggi di Lunghezza Arbitraria

La costruzione precedente funziona solo per messaggi di lunghezza fissa. Per autenticare messaggi di lunghezza arbitraria, potremmo pensare di suddividerli in blocchi e applicare il MAC a ciascun blocco. Tuttavia, questo approccio è insicuro.

- **Attacco di riordino:** Se il tag di $m = m_1 \,||\, m_2$ è $(t_1, t_2)$, un avversario può creare un tag valido $(t_2, t_1)$ per il nuovo messaggio $m' = m_2 \,||\, m_1$.
- **Attacco di troncamento:** Aggiungendo un indice di blocco si previene il riordino, ma un avversario può ancora rimuovere blocchi dalla fine del messaggio e del tag.
- **Attacco "mix-and-match":** Aggiungendo anche la lunghezza totale del messaggio in ogni blocco, un avversario può ancora combinare blocchi provenienti da messaggi diversi ma della stessa lunghezza per creare una nuova falsificazione.

### 29.2.1 Una Costruzione Sicura (Costruzione 4.7)

Per risolvere tutti questi problemi, si aggiunge a ogni blocco un **identificatore casuale del messaggio** $r$, che è unico per ogni nuovo messaggio da autenticare.

***CONSTRUCTION 4.7***

Let $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ be a fixed-length MAC for messages of length $n$. Define a MAC as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$, parse $m$ into $d$ blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Choose a uniform identifier $r \in \{0,1\}^{n/4}$.
  For $i = 1, \ldots, d$, compute $t_i \leftarrow \mathsf{Mac}'_k(r \| \ell \| i \| m_i)$, where $i, \ell$ are encoded as strings of length $n/4$.[†] Output the tag $t := \langle r, t_1, \ldots, t_d \rangle$.
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = \langle r, t_1, \ldots, t_{d'} \rangle$, parse $m$ into $d$ blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Output 1 if and only if $d' = d$ and $\mathsf{Vrfy}'_k(r \| \ell \| i \| m_i, t_i) = 1$ for $1 \leq i \leq d$.

[†] Note that $i$ and $\ell$ can be encoded using $n/4$ bits because $i, \ell < 2^{n/4}$.

Figura 5: Costruzione di un MAC per messaggi di lunghezza arbitraria (Costruzione 4.7).

Il tag per il blocco $m_i$ di un messaggio $m$ di lunghezza $L$ diventa:

$$t_i = \mathrm{Mac}'_k(r \,||\, L \,||\, i \,||\, m_i)$$

dove Mac$'$ è un MAC a lunghezza fissa. Il tag finale è $(r, t_1, ..., t_d)$.

**Teorema** (Teorema 4.8)**.** *Se $\Pi'$ è un MAC a lunghezza fissa sicuro, la Costruzione 4.7 è un MAC sicuro per messaggi di lunghezza arbitraria.*

**Dimostrazione** (Dimostrazione (del Teorema 4.8))**.** *Sia $\Pi$ lo schema MAC della Costruzione 4.7 e sia A un avversario PPT. La probabilità di successo di A, $Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1]$, può essere scomposta e maggiorata utilizzando la legge della probabilità totale. Introduciamo due eventi chiave:*

- ***Repeat**: L'evento che l'oracolo, nel rispondere alle query di A, generi due volte lo stesso identificatore casuale r per messaggi diversi.*
- ***Newblock**: L'evento che la falsificazione $(m, t = \langle r, t_1, ..., t_d\rangle)$ prodotta da A contenga almeno un blocco $b_i = r \,||\, L \,||\, i \,||\, m_i$ che non sia mai stato autenticato in precedenza (cioè, A non ha mai ricevuto il tag per $b_i$ tramite una query all'oracolo del MAC sottostante $\Pi'$).*

*Possiamo scrivere la probabilità di successo di A come:*

$$\begin{aligned} Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1] = {} & Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \textit{Repeat}] \\ & + Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\textit{Repeat}}] \end{aligned}$$

*Focalizzandoci sul secondo termine e introducendo l'evento Newblock:*

$$\begin{aligned} Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\textit{Repeat}}] = {} & Pr[\ldots \wedge \overline{\textit{Repeat}} \wedge \textit{Newblock}] \\ & + Pr[\ldots \wedge \overline{\textit{Repeat}} \wedge \overline{\textit{Newblock}}] \end{aligned}$$

*Da cui otteniamo la seguente maggiorazione:*

$$\begin{aligned} Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1] \leq {} & Pr[\textit{Repeat}] \\ & + Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \textit{Newblock}] \\ & + Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\textit{Repeat}} \wedge \overline{\textit{Newblock}}] \end{aligned}$$

*Analizziamo ora i tre termini separatamente.*

**Claim** (Claim 4.9 del libro di testo)**.** *$Pr[\textit{Repeat}]$ è trascurabile.*

**Dimostrazione.** *Se l'avversario A effettua $q(n)$ query all'oracolo, e l'identificatore casuale r è una stringa di $n/4$ bit scelta uniformemente, la probabilità che si verifichi una collisione (evento 'Repeat') è limitata superiormente dalla disuguaglianza del paradosso del compleanno:*

$$Pr[\textit{Repeat}] \leq \frac{q(n)(q(n)-1)}{2 \cdot 2^{n/4}} \approx \frac{q(n)^2}{2^{n/4+1}}$$

*Poiché $q(n)$ è un polinomio in n, questa probabilità è una funzione trascurabile.*

**Claim.** $Pr[\textit{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\textit{Repeat}} \wedge \overline{\textit{Newblock}}] = 0.$

**Dimostrazione.** *Questo termine rappresenta la probabilità che l'avversario vinca senza che si verifichi una ripetizione di $r$ e senza aver creato un nuovo "blocco atomico" da autenticare. Dimostriamo che se un avversario produce una falsificazione valida $(m, t)$ e non si è verificato 'Repeat', allora l'evento 'Newblock' **deve** essersi verificato. Sia $t = \langle r, t_1, \ldots, t_d \rangle$.*

- *Se $r$ è un identificatore "nuovo" (cioè non è mai stato restituito dall'oracolo nelle risposte alle query di A), allora tutti i blocchi $b_i = r \,||\, L \,||\, i \,||\, m_i$ sono per definizione nuovi, e quindi 'Newblock' si verifica.*
- *Se $r$ è un identificatore "vecchio", cioè $r = r_j$ per una qualche $j$-esima query fatta da $A$ sul messaggio $m_j$ di lunghezza $L_j$, allora, affinché la falsificazione sia valida, deve essere $m \neq m_j$. Analizziamo due sottocasi:*
  1. *Se la lunghezza $L$ di $m$ è diversa da $L_j$, allora ogni blocco $b_i = r \,||\, L \,||\, i \,||\, m_i$ è nuovo, perché il campo $L$ è diverso da $L_j$ presente in tutti i blocchi autenticati per la query $j$. Quindi, 'Newblock' si verifica.*
  2. *Se $L = L_j$, allora poiché $m \neq m_j$, deve esistere almeno un indice di blocco $i$ tale che $m_i \neq m_j^{(i)}$. Di conseguenza, il blocco $b_i = r \,||\, L \,||\, i \,||\, m_i$ è diverso dal corrispondente blocco autenticato per $m_j$, e quindi è un nuovo blocco. 'Newblock' si verifica.*

*In ogni scenario in cui l'avversario ha successo e non ci sono ripetizioni di $r$, l'evento 'Newblock' è inevitabile. Pertanto, l'evento congiunto con $\overline{Newblock}$ è impossibile e la sua probabilità è zero.*

**Claim** (Claim 4.10 del libro di testo)**.** *$Pr[Mac\text{-}forge_{A,\Pi}(n) = 1 \wedge Newblock]$ è trascurabile.*

**Dimostrazione.** *Questa affermazione si dimostra per riduzione. Si costruisce un avversario $A'$ che attacca il MAC a lunghezza fissa sottostante $\Pi'$. L'algoritmo $A'$ simula l'ambiente dell'esperimento Mac-forge per l'avversario $A$. Quando $A$ richiede un tag per un messaggio, $A'$ sceglie un $r$ casuale e, a sua volta, interroga il proprio oracolo $Mac'_k(\cdot)$ per ottenere i tag dei singoli blocchi, che poi assembla e restituisce ad $A$. Quando $A$ produce la sua falsificazione finale $(m, t)$, $A'$ verifica se l'evento 'Newblock' si è verificato. In caso affermativo, $A'$ estrae il primo blocco $(b, t_i)$ che risulta "nuovo" (cioè $b = r \,||\, L \,||\, i \,||\, m_i$ non è mai stato oggetto di una query da parte di $A'$ al suo oracolo) e lo restituisce come propria falsificazione contro $\Pi'$. Se l'evento $Mac\text{-}forge_{A,\Pi}(n) = 1 \wedge Newblock$ si verifica, significa che $A$ ha prodotto un tag valido per un messaggio nuovo che contiene almeno un blocco nuovo. In questo caso, $A'$ riesce a sua volta a produrre una falsificazione valida per $\Pi'$. Poiché $\Pi'$ è sicuro per ipotesi, la probabilità di successo di $A'$ è trascurabile. Di conseguenza, anche la probabilità dell'evento congiunto per $A$ deve essere trascurabile.*

*Poiché tutti e tre i termini della disuguaglianza iniziale sono trascurabili o nulli, la loro somma è trascurabile. Questo conclude la dimostrazione che la Costruzione 4.7 è sicura.*

# 30 Standard Pratici: CBC-MAC, GMAC e Poly1305

La costruzione precedente è pedagogicamente utile, ma inefficiente. Nella pratica si usano standard più performanti.

## 30.1 CBC-MAC

Il **CBC-MAC** è uno standard basato sulla modalità CBC. Per un messaggio $m = m_1 \,||\, ... \,||\, m_l$, il tag è l'output dell'ultima iterazione.

$$t_0 := 0^n, \quad t_i := F_k(t_{i-1} \oplus m_i) \quad \text{per } i = 1, ..., l$$

Il tag è $t_l$.

**Teorema** (Teorema 4.12)**.** *Se $F$ è una PRF, il CBC-MAC di base è un MAC sicuro per messaggi di* ***lunghezza fissa****.*

**Nota.** *Il CBC-MAC di base è insicuro se usato con messaggi di lunghezza variabile. Le varianti sicure includono:*

- *Preporre la lunghezza del messaggio prima di calcolare il MAC.*
- *Usare due chiavi: una per il CBC-MAC e una seconda per cifrare il tag finale: $t^* = F_{k_2}(t)$.*

## 30.2 GMAC e Poly1305

**GMAC** e **Poly1305** sono due standard moderni per l'autenticazione di messaggi, noti per la loro alta efficienza e per essere parallelizzabili. A differenza del CBC-MAC, che è intrinsecamente sequenziale, questi schemi si basano su un paradigma diverso che combina una **funzione $\epsilon$-difference universal** con una funzione pseudocasuale (PRF).

**Definizione** (Funzione $\epsilon$-difference universal - Definizione 4.14)**.** *Una funzione con chiave $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, dove $\mathcal{T}$ è un gruppo, è detta $\epsilon(n)$-**difference universal** se per ogni coppia di messaggi distinti $m, m' \in \mathcal{M}$ e per ogni $\Delta \in \mathcal{T}$, vale:*

$$Pr_{k \leftarrow \mathcal{K}}[h_k(m) - h_k(m') = \Delta] \leq \epsilon(n)$$

*dove la probabilità è calcolata sulla scelta uniforme della chiave $k$ da $\mathcal{K}$. Per questa definizione, deve necessariamente valere $\epsilon(n) \geq 1/ \mid \mathcal{T}_n \mid$.*

**Schema Generico.** La costruzione generale di un MAC basato su questa primitiva è randomizzata e funziona come segue:

1. **Gen:** Sceglie due chiavi indipendenti: $k_h \in \mathcal{K}$ per la funzione $h$ e $k_F \in \{0,1\}^n$ per una PRF $F$.

2. **Mac:** Per autenticare un messaggio $m$, sceglie un valore casuale (nonce) $r \in \{0,1\}^n$ e calcola il tag $t = \langle r, s \rangle$, dove $s = h_{k_h}(m) + F_{k_F}(r)$.

3. **Vrfy:** Data una coppia $(m, t = \langle r, s \rangle)$, verifica se $s = h_{k_h}(m) + F_{k_F}(r)$.

**Costruzione basata sui Polinomi.** Sia GMAC che Poly1305 istanziano la funzione $h$ usando una costruzione basata sulla valutazione di polinomi su un campo finito $\mathbb{F}$. L'idea è di interpretare un messaggio $m = m_1, \ldots, m_{t-1}$ come i coefficienti di un polinomio $m(X)$ e di valutare questo polinomio in un punto segreto, che è la chiave $k_h \in \mathbb{F}$[cite: 5562, 5563].

$$h_{k_h}(m) = m(k_h)$$

Si può dimostrare che questa costruzione è $\ell / \mid \mathbb{F} \mid$-difference universal, dove $\ell$ è il grado massimo del polinomio.

**Dimostrazione** (Dimostrazione (schizzo))**.** *Siano $m \neq m'$ due messaggi e sia $\Delta \in \mathbb{F}$ un valore qualsiasi. Vogliamo limitare la probabilità che $h_{k_h}(m) - h_{k_h}(m') = \Delta$. Definiamo un nuovo polinomio $P(X) = m(X) - m'(X) - \Delta$. Poiché $m \neq m'$, $P(X)$ è un polinomio non nullo di grado al più $\ell$. La condizione $h_{k_h}(m) - h_{k_h}(m') = \Delta$ è equivalente a $P(k_h) = 0$. Poiché un polinomio non nullo di grado $\ell$ su un campo ha al massimo $\ell$ radici (zeri), e la chiave $k_h$ è scelta uniformemente in $\mathbb{F}$, la probabilità che $k_h$ sia una di queste radici è al più $\ell / \mid \mathbb{F} \mid$.*

**Istanze Specifiche:**

- **GMAC (Galois/Counter Mode MAC):** Utilizza un cifrario a blocchi (come AES) per istanziare la PRF e definisce la funzione polinomiale sul campo finito $\mathbb{F}_{2^{128}}$.
- **Poly1305:** È una costruzione simile che usa il campo finito $\mathbb{F}_p$ dove $p$ è il numero primo $2^{130} - 5$. Il tag finale viene poi ridotto modulo $2^{128}$ per allinearsi con l'output della PRF, tipicamente istanziata con AES o ChaCha20.

Questi schemi sono particolarmente apprezzati per le loro elevate prestazioni e per le loro "strette" (tighter) riduzioni di sicurezza, che offrono garanzie concrete più forti rispetto a CBC-MAC a parità di parametri.

# 31 Lezione 12: Attacchi Chosen-Ciphertext (CCA) e Padding Oracle

In questa lezione, introduciamo una classe di attacchi più potente rispetto a quelli visti finora: gli **attacchi con cifrato scelto** (Chosen-Ciphertext Attacks - CCA). Questi attacchi modellano un avversario attivo che non solo osserva il canale, ma ha anche la capacità di ottenere la decifratura di cifrati di sua scelta.

## 31.1 Definizione di Sicurezza CCA

Formalizziamo questa nozione di sicurezza attraverso un esperimento che estende quello per la sicurezza CPA. All'avversario viene concesso l'accesso a un **oracolo di decifratura**, $Dec_k(\cdot)$, oltre a quello di cifratura, $Enc_k(\cdot)$.

**Esperimento di Indistinguibilità CCA $\mathbf{PrivK}^{\mathbf{cca}}_{\mathcal{A},\Pi}(n)$** Per uno schema di cifratura a chiave privata $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ e un avversario $\mathcal{A}$, l'esperimento si svolge come segue:

1. Il challenger genera una chiave $k \leftarrow \text{Gen}(1^n)$.
2. L'avversario $\mathcal{A}$ riceve $1^n$ e ha accesso agli oracoli $\text{Enc}_k(\cdot)$ e $\text{Dec}_k(\cdot)$. Dopo una fase di interrogazione, $\mathcal{A}$ produce una coppia di messaggi $m_0, m_1$ di uguale lunghezza.
3. Il challenger sceglie un bit casuale $b \leftarrow \{0,1\}$ e calcola il cifrato di sfida $c^* \leftarrow \text{Enc}_k(m_b)$.
4. $\mathcal{A}$ riceve $c^*$ e può continuare a interrogare entrambi gli oracoli, con un'unica restrizione: **non può chiedere la decifratura di $c^*$ stesso**. Alla fine, restituisce un bit $b'$.
5. L'output dell'esperimento è 1 se $b' = b$ (successo), altrimenti 0.

**Definizione (Sicurezza CCA)** Uno schema di cifratura a chiave privata $\Pi$ è detto **CCA-sicuro** (o ha cifrature indistinguibili rispetto ad un attacco di tipo chosen-ciphertext) se, per ogni avversario $\mathcal{A}$ PPT, esiste una funzione trascurabile $\text{negl}(\cdot)$ tale che:

$$\Pr[\text{PrivK}^{\text{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq \tfrac{1}{2} + \text{negl}(n)$$

dove la probabilità è calcolata sulla casualità della chiave, del bit $b$, dell'algoritmo di cifratura e dell'avversario stesso.

**Plausibilità degli Attacchi CCA** Gli attacchi CCA sono molto realistici in pratica:

- Un server che risponde con messaggi di errore diversi a seconda della validità di un cifrato agisce, di fatto, come un oracolo di decifratura parziale.
- In protocolli crittografici complessi, le parti oneste possono essere indotte a decifrare messaggi manipolati, agendo inconsapevolmente come oracoli per l'avversario.

Nessuno degli schemi CPA-sicuri visti finora (es. Costruzione 3.28, modalità CBC, CTR) è CCA-sicuro. Essi sono vulnerabili a semplici attacchi di manipolazione del cifrato, una proprietà nota come **malleabilità**. La sicurezza CCA implica la **non-malleabilità**, ovvero la proprietà per cui un avversario non può modificare un cifrato per ottenere un nuovo cifrato il cui messaggio sottostante sia correlato in modo prevedibile al messaggio originale.

## 31.2 Padding-Oracle Attack

Questo è un potente attacco CCA pratico che sfrutta le informazioni trapelate da un oracolo che rivela solo se il padding di un messaggio decifrato è corretto o meno. L'attacco è stato sferrato con successo contro implementazioni reali di protocolli come TLS.

**Contesto: Modalità CBC con Padding** In modalità CBC, se un messaggio non ha una lunghezza multipla di quella del blocco $L$, deve essere completato (padded). Uno standard comune è il PKCS#7:

- Se mancano $b$ byte per completare l'ultimo blocco ($1 \leq b \leq L$), si aggiungono $b$ byte, ciascuno con il valore esadecimale di $b$ (es. $0x04040404$ se $b = 4$).
- Se il messaggio è già un multiplo della lunghezza del blocco, si aggiunge un intero blocco di $L$ byte, ciascuno con il valore di $L$ (es. $0x10...10$ per blocchi di 16 byte).

Durante la decifratura, il ricevitore controlla la validità del padding: se non è corretto, restituisce un errore "bad padding". Questo comportamento crea un **oracolo di padding**.

**Meccanismo dell'Attacco** Supponiamo che l'avversario intercetti un cifrato CBC $c = \langle c_0, c_1, c_2 \rangle$, dove $c_0 = \text{IV}$. La decifratura del blocco $m_2$ è data da $m_2 = \text{Dec}_k(c_2) \oplus c_1$. L'idea chiave è che manipolando $c_1$, l'avversario può indurre modifiche controllate e prevedibili in $m_2$:

$$c_1' = c_1 \oplus \Delta \Rightarrow m_2' = m_2 \oplus \Delta$$

**Fase 1: Scoprire la lunghezza del padding** $b$ L'avversario modifica sequenzialmente i byte di $c_1$ da sinistra a destra (dal primo all'ultimo). Sia $c_1'$ la versione di $c_1$ con il primo byte modificato. L'avversario invia $\langle c_0, c_1', c_2 \rangle$ all'oracolo di decifratura.

- Se l'oracolo restituisce "bad padding", significa che la modifica ha corrotto un byte del padding. Poiché il primo byte di $c_1$ corrisponde al primo byte di $m_2$, ciò implica che il padding si estende per tutta la lunghezza del blocco. Quindi $b = L$.
- Se l'oracolo non restituisce errore, significa che il primo byte non fa parte del padding, quindi $b < L$. L'avversario ripete il processo modificando il secondo byte, e così via.

Il primo byte modificato (da sinistra) che causa un errore di padding rivela l'inizio del padding e quindi la sua lunghezza $b$.

**Fase 2: Scoprire i byte del messaggio** Una volta noto $b$, l'avversario sa che l'ultimo blocco decifrato $m_2$ ha la forma $\langle M_1, ..., M_{L-b-1}, B, \underbrace{0x0b, ..., 0x0b}_{b \text{ volte}} \rangle$, dove $B$ è l'ultimo

byte del messaggio in chiaro che vuole scoprire. L'obiettivo è creare un nuovo $c_1'$ tale che il blocco decifrato $m_2'$ termini con un padding valido di lunghezza $b+1$, ovvero con $b+1$ byte di valore $b+1$. Per fare ciò, l'avversario calcola una maschera $\Delta_i$ per ogni possibile valore $i \in \{0, ..., 255\}$ dell'ultimo byte del messaggio $B$. La maschera è costruita per trasformare gli ultimi $b+1$ byte di $m_2$ in $... \mid\mid 0x(b+1) \mid\mid \underbrace{0x(b+1), ..., 0x(b+1)}_{b \text{ volte}}$.

L'avversario invia $c_1' = c_1 \oplus \Delta_i$ per $i = 0, 1, ...$ fino a quando l'oracolo non segnala un padding valido. Quando ciò accade, l'avversario sa che l'ultimo byte di $m_2'$ è $0x(b+1)$, e da questa informazione può risalire al valore di $B$. Questo processo viene iterato per recuperare tutti i byte del messaggio.

# 32 Lezione 13: Cifratura Autenticata

Per ottenere simultaneamente confidenzialità e integrità in presenza di un avversario attivo, abbiamo bisogno di **schemi di cifratura autenticata** (Authenticated Encryption - AE).

## 32.1 Definizione di Cifratura Autenticata

Uno schema AE deve soddisfare due requisiti: essere CCA-sicuro (per la confidenzialità) e garantire l'integrità dei dati (non falsificabilità).

**Esperimento di Non-Falsificabilità Enc-Forge$_{\mathcal{A},\Pi}(n)$** Questo esperimento formalizza l'integrità per uno schema di cifratura.

1. Viene generata una chiave $k \leftarrow \text{Gen}(1^n)$.
2. L'avversario $\mathcal{A}$ riceve accesso all'oracolo $\text{Enc}_k(\cdot)$ e, dopo aver fatto delle query, produce un cifrato $c$. Sia $Q$ l'insieme dei messaggi che $\mathcal{A}$ ha inviato all'oracolo.
3. Sia $m := \text{Dec}_k(c)$. L'avversario ha successo (output 1) se e solo se $m \neq \perp$ e $m \notin Q$.

**Definizione (Cifratura Autenticata)** Uno schema di cifratura a chiave privata $\Pi$ è uno schema di **cifratura autenticata** se è CCA-sicuro e non falsificabile (unforgeable).

## 32.2 Approcci Generici alla Composizione

Dati uno schema di cifratura CPA-sicuro $\Pi_E$ e un MAC fortemente sicuro $\Pi_M$, esistono tre modi naturali per combinarli:

1. **Encrypt-and-authenticate**: Si cifrano e si autenticano i dati in parallelo: $c \leftarrow \text{Enc}_{k_E}(m)$, $t \leftarrow \text{Mac}_{k_M}(m)$. Il trasmesso è $\langle c, t \rangle$. **Insicuro**: Il MAC potrebbe rivelare informazioni sul messaggio (es. se è deterministico, rivela se lo stesso messaggio è stato inviato due volte), violando la sicurezza CPA della composizione.

2. **Authenticate-then-encrypt**: Prima si calcola il tag, poi si cifra il messaggio concatenato al tag: $t \leftarrow \text{Mac}_{k_M}(m)$, $c \leftarrow \text{Enc}_{k_E}(m \mid\mid t)$. **Insicuro**: Questo approccio è vulnerabile ad attacchi CCA. Se lo schema di cifratura sottostante è CBC con padding, un padding-oracle attack può essere sferrato. L'attaccante può distinguere tra un errore di padding e un errore di verifica del tag, ottenendo le informazioni necessarie per l'attacco.

3. **Encrypt-then-authenticate**: Prima si cifra il messaggio, poi si calcola il tag sul cifrato: $c \leftarrow \text{Enc}_{k_E}(m)$, $t \leftarrow \text{Mac}_{k_M}(c)$. Il trasmesso è $\langle c, t \rangle$. **Sicuro**: Questo è l'unico approccio genericamente sicuro.

**Costruzione Encrypt-then-authenticate** Sia $\Pi_E = (\text{Enc}, \text{Dec})$ uno schema di cifratura CPA-sicuro e $\Pi_M = (\text{Mac}, \text{Vrfy})$ un MAC fortemente sicuro.

- **Gen'**: Sceglie due chiavi indipendenti e uniformi $k_E, k_M \in \{0,1\}^n$.
- **Enc'**: Su input $(k_E, k_M)$ e $m$, calcola $c \leftarrow \text{Enc}_{k_E}(m)$ e $t \leftarrow \text{Mac}_{k_M}(c)$. L'output è $\langle c, t \rangle$.
- **Dec'**: Su input $(k_E, k_M)$ e $\langle c, t \rangle$, verifica se $\text{Vrfy}_{k_M}(c, t) = 1$. Se sì, restituisce $\text{Dec}_{k_E}(c)$; altrimenti, restituisce $\perp$.

**Teorema** Se $\Pi_E$ è CPA-sicuro e $\Pi_M$ è fortemente sicuro, la costruzione Encrypt-then-authenticate è uno schema di cifratura autenticata.

*Dimostrazione (schizzo)*:

1. **Non-falsificabilità**: Un avversario che cerca di creare un cifrato falsificato $\langle c', t' \rangle$ deve, per definizione, creare un tag $t'$ valido per un $c'$ che non ha ricevuto dall'oracolo. Poiché $\Pi_M$ è fortemente sicuro, la probabilità di riuscirci è trascurabile. Questo rende l'oracolo di decifratura inutile per un avversario CCA, poiché qualsiasi cifrato nuovo che costruisce verrà rifiutato dalla verifica del MAC con probabilità quasi 1.

2. **Sicurezza CCA**: La prova formale (Claim 4.20 nel libro di testo) mostra che la probabilità di un evento ValidQuery (un avversario che invia un cifrato nuovo e valido all'oracolo di decifratura) è trascurabile. Poiché l'oracolo di decifratura è reso inutile, un attacco CCA contro la costruzione si riduce a un attacco CPA contro lo schema $\Pi_E$ sottostante. Poiché $\Pi_E$ è CPA-sicuro per ipotesi, anche la costruzione combinata lo è.

**Necessità di Chiavi Indipendenti** È cruciale utilizzare chiavi indipendenti per la cifratura ($k_E$) e l'autenticazione ($k_M$). Se si usa la stessa chiave, possono emergere interazioni insicure tra le due primitive, anche se singolarmente sicure. Ad esempio, se si usa $F_k(m \mid\mid r)$ per cifrare e $F_k^{-1}(c)$ per autenticare, la composizione può rivelare il messaggio in chiaro.

## 32.3 Standard Pratici

Diversi schemi di cifratura autenticata standardizzati si basano su queste idee:

- **GCM (Galois/Counter Mode)**: Usa l'approccio Encrypt-then-authenticate, combinando la modalità CTR per la cifratura con il MAC GMAC per l'autenticazione. È molto efficiente, specialmente su hardware con supporto AES-NI.
- **CCM (Counter with CBC-MAC)**: Usa l'approccio Authenticate-then-encrypt, combinando CTR e CBC-MAC. È più lento e meno flessibile di GCM, ma richiede solo un cifrario a blocchi per l'implementazione.
- **ChaCha20-Poly1305**: Usa Encrypt-then-authenticate, combinando lo stream cipher ChaCha20 con il MAC Poly1305. Offre prestazioni eccellenti in software ed è un'alternativa a GCM su piattaforme senza accelerazione hardware.

Tutti e tre questi schemi sono inclusi nello standard TLS 1.3 e possono supportare dati associati (AEAD) che vengono autenticati ma non cifrati.