Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

SICUREZZA ASSOLUTA E COMPUTAZIONALE DI UN SISTEMA CRITTOGRAFICO A CHIAVE PRIVATA

Tesi di Laurea in Algoritmi della Teoria dei Numeri e Crittografia

Relatore:
Chiar.mo Prof.
DAVIDE ALIFFI

Presentata da:
VALENTINA GUERRA

Sessione I
Anno Accademico 2012/2013

Alla mia famiglia...

Introduzione

Il primo passo per risolvere un problema crittografico è la formulazione di una rigorosa definizione di sicurezza. In questo elaborato andremo prima di tutto a definire il concetto di sicurezza e vedremo sotto quali condizioni un sistema può essere considerato sicuro. A tal proposito vedremo che si è soliti distinguere tra due tipi di sicurezza:

- *Sicurezza perfetta* quando un sistema si dimostra sicuro contro avversari dotati di illimitate risorse computazionali.
- *Sicurezza computazionale* quando un avversario potrebbe violare il sistema avendo a disposizione "abbastanza" tempo (tanto da rendere praticamente trascurabile la probabilità di tale evento).

La costruzione di sistemi computazionalmente sicuri è necessaria poiché questi sono efficienti e mantengono un livello accettabile di sicurezza, mentre i sistemi a sicurezza perfetta risultano inefficienti dal punto di vista pratico. Vedremo il One-Time Pad (cifrario di Vernam), un sistema crittografico che ha sicurezza perfetta. La prima dimostrazione della sua inviolabilità fu pubblicata nel 1949 da Claude Shannon nell'articolo *Communication Theory of Secrecy Systems*[3]. Infine andremo a costruire un sistema computazionalmente sicuro, simile al One-Time Pad, ma efficiente da un punto di vista pratico.

Indice

Capitolo 1

Schemi a sicurezza perfetta

In questo capitolo studieremo schemi di cifratura che sono sicuri contro avversari con illimitate risorse computazionali. Tali schemi sono detti *a sicurezza perfetta*.

1.1 Premesse

Innanzitutto definiamo un generico sistema a chiave simmetrica. Supponiamo che due parti debbano comunicare segretamente tra loro, per semplicità chiamiamo Alice e Bob i due interlocutori. Per prima cosa, Alice genera una chiave che viene trasmessa a Bob, in modo tale da non essere intercettata. Utilizzando la chiave, Alice cifra il messaggio che deve inviare a Bob e lo trasmette. Arrivato a destinazione, il messaggio viene decifrato da Bob tramite la chiave. In questo scenario la stessa chiave viene usata per convertire il testo in chiaro in testo cifrato e viceversa: per questo motivo questo tipo di cifrari vengono detti a chiave simmetrica. Le due parti devono però riuscire a scambiarsi la chiave senza che questa venga intercettata. Uno schema di cifratura a chiave simmetrica consiste dunque di tre algoritmi: il primo è una procedura per generare chiavi, il secondo una procedura per cifrare, il terzo una procedura per decifrare.

Vediamoli nel dettaglio:

1. L'algoritmo che genera la chiave, **Gen**, è un algoritmo probabilistico che restituisce una chiave k secondo una distribuzione che dipende dallo schema (di solito si suppone che la distribuzione sia uniforme).

2. L'algoritmo di cifratura **E** prende in input una chiave k e un testo in chiaro m e restituisce un testo cifrato c. Scriviamo $E_k(m)$ per indicare la cifratura del testo in chiaro m tramite la chiave k.

3. L'algoritmo per decifrare **D** prende in input una chiave k e un testo cifrato c e restituisce un testo in chiaro m. Indichiamo con $D_k(c)$ la decifrazione del testo cifrato c tramite la chiave k.

Il seguente schema, simile a quello apparso nell'articolo di Shannon *Communication Theory of Secrecy Systems*[3], riassume quanto detto.

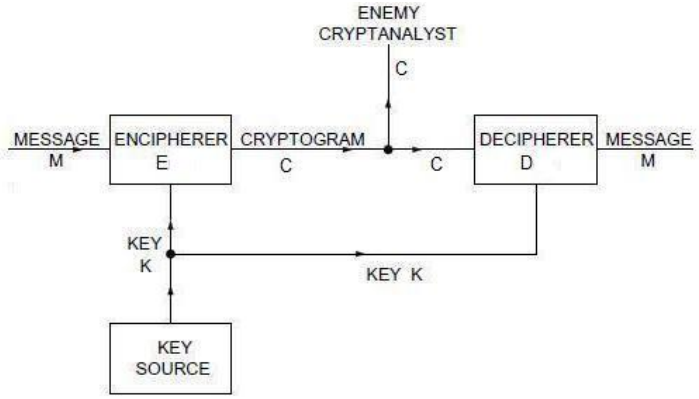

Figura 1.1: Schema di un generico sistema a chiave simmetrica

L'insieme di tutte le possibili chiavi generate dall'algoritmo *Gen* è chiamato *spazio delle chiavi* e si indica con $\mathcal{K}$. L'insieme di tutti i possibili messaggi è chiamato *spazio dei testi in chiaro* e si denota con $\mathcal{M}$. Infine indichiamo con $\mathcal{C}$ l'insieme di tutti i possibili testi cifrati c generati da $E_k(m)$ per tutte le possibili scelte di $k \in \mathcal{K}$ e $m \in \mathcal{M}$ e lo chiamiamo *spazio dei testi cifrati*. Dunque $\mathcal{C}$ è definito da $\mathcal{K}$ e $\mathcal{M}$.

Possiamo affermare che uno schema di cifratura è definito dai tre algoritmi (Gen, E, D) e dallo spazio $\mathcal{M}$ e si ha che, per ogni chiave k generata da Gen e ogni messaggio $m \in \mathcal{M}$,

$$D_k(E_k(m)) = m$$

Si nota dalla discussione precedente che, se un avversario viene in possesso della chiave e dell'algoritmo di decifrazione, sarà poi in grado di decifrare l'intera comunicazione delle parti. Già dall'ottocento, tuttavia, i crittografi si sono resi conto che mantenere segreto il metodo usato per cifrare non è un buon modo per garantire la sicurezza delle comunicazioni.

Principio di Kerckhoffs. *Non deve essere necessario tener segreto il metodo di cifratura, e non ci devono essere inconvenienti se questo cade nelle mani dei nemici.*

Un sistema deve quindi rimanere sicuro anche quando il metodo di cifratura viene reso pubblico, dunque è richiesta solo la segretezza della chiave. Adottare questo principio porta a dei sostanziali vantaggi: innanzitutto, è più facile tener segreta solo la chiave, piuttosto che un intero algoritmo, e se la chiave viene scoperta, basta cambiarla.
Oggi il principio di Kerckhoffs viene inteso nel senso che l'algoritmo *deve* essere completamente pubblico. In questo modo l'algoritmo può essere studiato dalla comunità dei crittografi e se presenta delle debolezze queste verranno scoperte più facilmente.

1.1.1 Richiami di probabilità

Sia X una variabile aleatoria che assume valori in un insieme discreto Ω e sia $x \in \Omega$, indichiamo con $P(X = x)$ la probabilità che X assuma il valore x.

Definizione 1.1. Siano X, Y due variabili aleatorie. Si chiama allora **Probabilità congiunta**

$$P(X = x, Y = y)$$

la probabilità che i due eventi $(X = x)$ e $(Y = y)$ si verifichino contemporaneamente.

Osservazione 1. Se due variabili aleatorie X, Y sono indipendenti, si ha

$$P(X = x, Y = y) = P(X = x)P(Y = y)$$

Definizione 1.2. Siano X, Y due variabili aleatorie e sia $P(X = x) > 0$. Si chiama **Probabilità condizionata dell'evento** $(Y = y)$ **rispetto all'evento** $(X = x)$ la probabilità che si verifichi l'evento $(Y = y)$, sapendo che si è verificato l'evento $(X = x)$. Si ha che:

$$P(Y = y|X = x) = \frac{P(X = x, Y = y)}{P(X = x)}$$

Teorema 1.1.1 (Formula di Bayes)**.** *Siano X, Y due variabili aleatorie e sia $P(X = x) > 0$. Allora vale la formula di Bayes*

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{P(X = x)}$$

Teorema 1.1.2 (Formula della probabilità totale)**.** *Siano X una variabile aleatoria e $Y_1, ..., Y_n$ una partizione di eventi. Allora*

$$P(X = x) = \sum_{i=1}^{n} P(X = x|Y_i)P(Y_i)$$

1.1.2 Entropia

Nella teoria dell'informazione l'entropia è una grandezza che definisce il livello di incertezza a cui è soggetta una variabile aleatoria, ad esempio l'entropia associata ad un messaggio m quantifica l'informazione che si ottiene da m all'uscita del canale di comunicazione. Lo studio dell'entropia nella teoria dell'informazione è stato affrontato per la prima volta da Claude Shannon nell'articolo *A Mathematical Theory of Communication* del 1948[4]. Vediamone alcuni risultati.

Definizione 1.3. Data una variabile aleatoria discreta X con distribuzione di probabilità $p_1, p_2, ..., p_n$, l'entropia H è data da

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

e definisce la quantità media di informazione contenuta in X.

Definizione 1.4. Date due variabili aleatorie X, Y, si definisce **Entropia condizionata**

$$H(Y|X) = -\sum_{i,j} p_i p_i(j) \log_2 p_i(j)$$

dove, con $p_i(j)$, intendiamo la probabilità condizionata $P(Y = y_j|X = x_i)$.

Definizione 1.5. Date due variabili aleatorie X, Y, si dice **Entropia congiunta**

$$H(X, Y) = -\sum_{i,j} p(i, j) \log_2 p(i, j)$$

dove, con $p(i, j)$, intendiamo la probabilità congiunta $P(X = x_i, Y = y_j)$.

Osservazione 2. Si verifica che

- $H(X, Y) = H(X) + H(Y|X)$
- $H(X, Y) = H(X) + H(Y) \Leftrightarrow X$ e Y sono indipendenti
- $H(X|Y) = H(X) \Leftrightarrow X$ e Y sono indipendenti

1.2 Sicurezza perfetta

Riprendendo quanto detto nelle premesse, uno schema di cifratura è definito dai tre algoritmi (Gen, E, D) e dallo spazio dei messaggi $\mathcal{M}$:

- Gen è un algoritmo probabilistico che genera una chiave k seguendo una certa distribuzione, solitamente la distribuzione uniforme. Supponiamo che lo spazio delle chiavi $\mathcal{K}$ sia finito.

- L'algoritmo di cifratura E potrebbe essere probabilistico, e in tal caso $E_k(m)$ non produce ogni volta lo stesso testo cifrato per un dato testo in chiaro m e una data chiave k. Nel caso invece in cui E sia deterministico, ad ogni k ed m è associato un solo testo cifrato e possiamo scrivere $c := E_k(m)$.

- Assumiamo invece che D sia deterministico, cioè che per ogni $k \in \mathcal{K}$, $m \in \mathcal{M}$ e ogni c generato da $E_k(m)$, si ha $D_k(c) = m$; in tal caso decifrare il testo c con la chiave k dà sempre il testo m. Se così non fosse, Bob non potrebbe decifrare correttamente il testo c, o meglio otterrebbe un testo in chiaro che potrebbe non corrispondere al testo scelto da Alice.[1]

Consideriamo ora le variabili aleatorie K, M e C, che assumono valori rispettivamente negli spazi $\mathcal{K}$, $\mathcal{M}$ e $\mathcal{C}$, e le relative distribuzioni di probabilità. Con $P(K = k)$ intendiamo la probabilità che la chiave generata da Gen sia la chiave k, similmente $P(M = m)$ è la probabilità che il messaggio sia m, con $m \in \mathcal{M}$, e $P(C = c)$ è la probabilità che il messaggio cifrato sia c. Le distribuzioni di K e M sono supposte essere indipendenti poiché normalmente la chiave e il messaggio sono scelti indipendentemente, mentre, dato l'algoritmo E, la distribuzione di C è determinata da quelle di K e M.[2]
Diamo ora la nozione di *sicurezza perfetta*. Immaginiamo un avversario con illimitate capacità computazionali che conosca la distribuzione di probabilità di M, e che venga in possesso di un testo cifrato. Allora osservare il testo cifrato non dovrebbe fornire all'avversario alcuna conoscenza sul testo in chiaro. In altre parole, la probabilità a posteriori di un messaggio m deve coincidere con quella a priori. Formalmente:

Definizione 1.6. Un cifrario simmetrico (Gen, E, D) su uno spazio $\mathcal{M}$ ha **sicurezza perfetta** se per ogni distribuzione di probabilità su $\mathcal{M}$, ogni

[1] Esistono comunque sistemi simmetrici in cui la decifrazione non è deterministica, ad esempio NTRU.

[2] La distribuzione di M dipende da molti fattori, ad esempio la lingua utilizzata.

$m \in \mathcal{M}$, e ogni $c \in \mathcal{C}$ per cui $P(C = c) > 0$:

$$P(M = m|C = c) = P(M = m)$$

cioè la probabilità a priori sul messaggio m rimane inalterata dopo aver osservato il testo cifrato c.[3]
Il seguente Lemma fornisce alcune formulazioni equivalenti della Definizione.

Lemma 1.2.1. *Sono equivalenti:*

(i) *Per ogni distribuzione su $\mathcal{M}$, ogni $m \in \mathcal{M}$, e ogni $c \in \mathcal{C}$:*

$$P(M = m|C = c) = P(M = m)$$

(ii) *Per ogni distribuzione su $\mathcal{M}$, ogni $m \in \mathcal{M}$, e ogni $c \in \mathcal{C}$:*

$$P(C = c|M = m) = P(C = c)$$

(iii) *Per ogni distribuzione su $\mathcal{M}$, ogni $m_0, m_1 \in \mathcal{M}$, e ogni $c \in \mathcal{C}$:*

$$P(C = c|M = m_0) = P(C = c|M = m_1)$$

La (i) è la definizione di sicurezza perfetta, la (ii) afferma che la distribuzione di probabilità di C è indipendente dal testo in chiaro[4]. La (iii) invece equivale a dire che, dati due testi in chiaro m_0 e m_1, è impossibile distinguere tra la cifratura dei due.[5]

Dimostrazione. Dimostriamo che $(ii) \Rightarrow (i) \Rightarrow (iii) \Rightarrow (ii)$.
$(ii) \Rightarrow (i)$ Fissiamo una distribuzione su $\mathcal{M}$, un $m \in \mathcal{M}$ e un $c \in \mathcal{C}$. Allora per ipotesi

$$P(C = c|M = m) = P(C = c)$$

[3]Da qui in poi supporremmo sempre che $P(C = c) > 0$ e $P(M = m) > 0$ per semplicità di esposizione.

[4]Questo non accade per cifrari con sicurezza non assoluta.

[5]Questa nozione viene ripresa come "proprietà di indistinguibilità" nel contesto della sicurezza computazionale.

Moltiplicando ambo i membri per $\frac{P(M=m)}{P(C=c)}$ otteniamo

$$\frac{P(C=c|M=m)P(M=m)}{P(C=c)} = P(M=m)$$

Per il teorema di Bayes, il fattore a sinistra dell'equazione è esattamente pari a $P(M=m|C=c)$. Sostituendo

$$P(M=m|C=c) = P(M=m)$$

$(i) \Rightarrow (iii)$ Supponiamo che $P(M=m|C=c) = P(M=m)$ e procedendo come prima moltiplichiamo ambo i membri per $\frac{P(C=c)}{P(M=m)}$. Otteniamo

$$\frac{P(M=m|C=c)P(C=c)}{P(M=m)} = P(C=c)$$

quindi, per il teorema di Bayes

$$P(C=c|M=m) = P(C=c)$$

Ora, fissiamo $m_0, m_1 \in \mathcal{M}$ e $c \in \mathcal{C}$. Per quanto appena dimostrato,

$$P(C=c|M=m_0) = P(C=c) = P(C=c|M=m_1)$$

$(iii) \Rightarrow (ii)$ Supponiamo che per ogni distribuzione su $\mathcal{M}$, ogni $m_0, m_1 \in \mathcal{M}$, e ogni $c \in \mathcal{C}$ sia

$$P(C=c|M=m_0) = P(C=c|M=m_1)$$

Fissiamo una distribuzione su $\mathcal{M}$, un $m_0 \in \mathcal{M}$ e un $c \in \mathcal{C}$. Definiamo $p = P(C=c|M=m_0)$.
Sappiamo che $P(C=c|M=m) = P(C=c|M=m_0) = p$ per tutti gli

$m \in \mathcal{M}$, allora

$$\begin{aligned} P(C = c) &= \sum_{m \in M} P(C = c | M = m) \cdot P(M = m) \\ &= \sum_{m \in M} p \cdot P(M = m) \\ &= p \cdot \sum_{m \in M} P(M = m) \\ &= p \\ &= P(C = c | M = m_0) \end{aligned}$$

Per l'arbitrarietà di m_0, abbiamo dimostrato che per ogni $c \in \mathcal{C}$ e $m \in \mathcal{M}$ $P(C = c) = P(C = c|M = m)$. □

In termini di entropia, si ha che un sistema crittografico è perfetto se e solo se verifica la seguente condizione:

$$H(M|C) = H(M)$$

1.3 Il One-Time Pad

Nel 1917, Gilbert Vernam ideò un cifrario, chiamato One-Time Pad, che si rivelò poi essere perfettamente sicuro. Nel One-Time Pad il messaggio viene rappresentato tramite una sequenza di bit uguali a 1 o a 0, mentre la chiave viene generata come una sequenza casuale di bit della stessa lunghezza del messaggio. Supponiamo ad esempio che messaggio, chiave e testo cifrato siano di lunghezza l, con l intero positivo. Si ha dunque che $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^l$. Poichè la chiave viene scelta da *Gen* casualmente, seguendo la distribuzione unifome, ogni chiave ha la stessa probabilità $\frac{1}{2^l}$ di essere scelta. Per cifrare si sommano bit a bit mod 2 il messaggio e la chiave. Questa operazione viene chiamata XOR (Exclusive Or) e restituisce 1 se e solo se uno solo degli operandi è uguale a 1:
$0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 0$.
Ad esempio[2], se il messaggio è $m = 00101001$ e la chiave è $k = 10101100$

otteniamo il seguente testo cifrato $c := m \oplus k$:

$$\begin{array}{lll} (m) & 00101001 & \oplus \\ (k) & 10101100 & = \\ (c) & 10000101 & \end{array}$$

Per decifrare si utilizza la stessa chiave, semplicemente aggiungendola al testo cifrato tramite l'operazione XOR: $10000101 \oplus 10101100 = 00101001$ ($c \oplus k$=m). Una variazione consiste nel mantenere il testo in chiaro in lettere alfabetiche e, immaginando di numerare le lettere da 0 a 25 (A=0,...,Z=25), sommare la chiave mod 26 ($c := m + k \mod 26$). In questo caso la chiave sarebbe quindi una sequenza casuale di lettere. Per decifrare si fa la stessa cosa, ma la chiave viene sottratta invece che aggiunta ($m = c - k \mod 26$). Ad esempio cifrando il messaggio "attaccare" tramite la chiave JUFNRBKAQ otteniamo il messaggio cifrato JNYNTDKRU. Se l'avversario ha a disposizione un solo testo cifrato, questo sistema risulta inviolabile. Supponiamo ad esempio[2] che c sia FIOWPSLQNTISJQL, il messaggio allora potrebbe essere wewillwinthewar o qualsiasi altro messaggio della stessa lunghezza. Questo perchè il testo cifrato non fornisce alcuna informazione riguardo il testo in chiaro, tranne la sua lunghezza. Se conosciamo una parte del testo in chiaro, allora conosciamo anche la parte corrispondente della chiave, ma questo non ci aiuterà a decifrare il resto.
Shannon nel 1949 dimostrò la sicurezza del cifrario di Vernam nel suo celebre articolo *Communication Theory of Secrecy Systems*[3]. Si ha dunque il seguente teorema:

Teorema 1.3.1. *Il cifrario di Vernam ha segretezza perfetta.*

Dimostrazione 1. Fissata una distribuzione su $\mathcal{M}$, un $m \in \mathcal{M}$ e un $c \in \mathcal{C}$, si ha che

$$\begin{aligned} P(C = c | M = m) &= P(M \oplus K = c | M = m) \\ &= P(m \oplus K = c) \\ &= P(K = m \oplus c) \\ &= \frac{1}{2^l} \end{aligned}$$

Poichè questo vale per tutte le distribuzioni e per tutti gli m, si ha che per ogni distribuzione di probabilità su $\mathcal{M}$, ogni $m_0, m_1 \in \mathcal{M}$ e ogni $c \in C$,

$$P(C = c|M = m_0) = \frac{1}{2^l} = P(C = c|M = m_1)$$

dunque per il punto (iii) del Lemma 1.2.1, lo schema è perfettamente sicuro.

□

Dimostriamo ora il teorema utilizzando il concetto di entropia

Dimostrazione 2. Dobbiamo provare che $H(M|C) = H(M)$.
Siano $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^l$ e sia $c \in \mathcal{C}$, allora

$$\begin{aligned} P(C = c) &= \sum_{m,k,E_k(m)=c} P(M = m) \cdot P(K = k) \\ &= \sum_{m,k,E_k(m)=c} P(M = m) \cdot \frac{1}{2^l} \\ &= \frac{1}{2^l} \sum_{m,k,E_k(m)=c} P(M = m) \end{aligned}$$

Osservo che $\forall m \in M$, $\forall c \in C$, $\exists! k \in K$ tale che $E_k(m) = c$, infatti

$$E_k(m) = m \oplus k = c \Rightarrow k = c \oplus m$$

Dunque

$$\begin{aligned} P(C = c) &= \frac{1}{2^l} \sum_{m,k,E_k(m)=c} P(M = m) \\ &= \frac{1}{2^l} \sum_{m \in \mathcal{M}} P(M = m) \\ &= \frac{1}{2^l} \end{aligned}$$

Si ha così che

$$H(C) = -\sum_{c \in C} P(C = c) \log_2 P(C = c) = -\sum_{c \in C} \frac{1}{2^l}(-l) = l$$

Poichè $\mathcal{K} = \mathcal{C} = \{0,1\}^l$ e $P(K = k) = \frac{1}{2^l} = P(C = c)$, allora vale anche $H(K) = H(C) = l$. Osserviamo ora che $H(M, C) = H(M, K)$, poichè fissato

il testo in chiaro l'incertezza sul testo cifrato è uguale a quella sulle chiavi, e utilizziamo la relazione $H(M,C) = H(M|C) + H(C)$. Ne verrà che

$$\begin{aligned} &H(M,K) = H(M|K) + H(K) = H(M) + H(K) \\ &\Rightarrow H(M|C) + H(C) = H(M) + H(K) \\ &\Rightarrow H(M|C) = H(M) \end{aligned}$$

□

Il cifrario di Vernam ha tuttavia degli svantaggi. Primo fra tutti, la chiave deve essere lunga quanto il messaggio e dunque sarà più difficile da generare, scambiare e tenere al sicuro. Inoltre, il cifrario rimane sicuro solo se ogni chiave viene utilizzata una sola volta, si rende dunque necessario generare e scambiare una chiave diversa ad ogni comunicazione. Dimostriamo ora brevemente cosa accadrebbe utilizzando due volte la stessa chiave.

Osservazione 3. Siano m_1, m_2 due messaggi in chiaro e siano $c_1 = m_1 \oplus k$ e $c_2 = m_2 \oplus k$ i corrispondenti testi cifrati con la stessa chiave k. Un avversario che abbia ottenuto c_1 e c_2 potrebbe calcolare $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$ $= m_1 \oplus m_2$ venendo a conoscenza dello XOR tra i due messaggi, ottenendo così una conoscenza, anche se parziale, di m_1 e m_2.

Si dimostra col seguente teorema che qualsiasi cifrario a sicurezza perfetta deve avere lo spazio delle chiavi grande almeno quanto quello dei messaggi in chiaro, e dunque la chiave stessa deve essere lunga almeno quanto il messaggio.

Teorema 1.3.2. *Sia (Gen, E, D) uno schema a sicurezza perfetta definito su $\mathcal{M}$, e sia $\mathcal{K}$ lo spazio delle chiavi generate da Gen. Allora*

$$|\mathcal{K}| \geq |\mathcal{M}|$$

Dimostrazione. Dimostriamo che se $|\mathcal{K}| < |\mathcal{M}|$ allora lo schema non è perfettamente sicuro. Assumiamo che $|\mathcal{K}| < |\mathcal{M}|$ e consideriamo la distribuzione uniforme su $\mathcal{M}$. Sia $c \in \mathcal{C}$ e sia

$$\mathcal{M}(c) := \{\widehat{m} | \widehat{m} = D_{\widehat{k}}(c), \widehat{k} \in \mathcal{K}\}$$

. Chiaramente $|\mathcal{M}(c)| \leq |\mathcal{K}|$ poichè per ogni messaggio $\widehat{m} \in \mathcal{M}(c)$ esiste almeno un $\widehat{k} \in \mathcal{K}$ per cui $\widehat{m} = D_{\widehat{k}}(c)$ per come abbiamo definito $\mathcal{M}(c)$. Poichè abbiamo assunto $|\mathcal{K}| < |\mathcal{M}|$, allora esiste almeno un $m' \in \mathcal{M}$ tale che $m' \notin \mathcal{M}(c)$. Ma allora

$$P(M = m'|C = c) = 0 \neq P(M = m')$$

e dunque lo schema non è perfettamente sicuro. □

1.4 Teorema di Shannon

Per concludere questo capitolo, vediamo come Shannon è riuscito a caratterizzare gli schemi a sicurezza perfetta. Supponendo che $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$, allora *Gen* deve scegliere una chiave *uniformemente* dallo spazio delle chiavi e per ogni messaggio in chiaro e per ogni testo cifrato deve esistere una sola chiave che trasforma il messaggio in chiaro in testo cifrato.

Teorema 1.4.1 (Teorema di Shannon)**.** *Sia* (Gen, E, D) *uno schema definito sullo spazio* $\mathcal{M}$ *tale che* $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$*. Lo schema è a sicurezza perfetta se e solo se:*

1. *Ogni chiave* $k \in \mathcal{K}$ *è scelta dall'algoritmo Gen con probabilità* $\frac{1}{|\mathcal{K}|}$*.*
2. $\forall m \in \mathcal{M}, \forall c \in \mathcal{C}$*,* $\exists! k \in \mathcal{K}$ *tale che* $E_k(m) = c$*.*

Dimostrazione. Supponiamo per semplicità che E sia deterministico.
($\Rightarrow$ 2.) Supponiamo che (Gen, E, D) sia perfettamente sicuro e dimostriamo il punto 2. Per ogni $m \in \mathcal{M}$ e $c \in \mathcal{C}$ esiste almeno una chiave $k \in \mathcal{K}$ tale che $E_k(m) = c$, altrimenti avremmo $P(M = m|C = c) = 0 \neq P(M = m)$.
Fissato m, consideriamo ora l'insieme $\{E_k(m)\}_{k\in\mathcal{K}}$.
Si ha che $|\{E_k(m)\}_{k\in\mathcal{K}}| \leq |\mathcal{C}|$, ma vale anche $|\{E_k(m)\}_{k\in\mathcal{K}}| \geq |\mathcal{C}|$ perchè $\forall c \in \mathcal{C}\ \exists k \in \mathcal{K}$ tale che $E_k(m) = c$.
Dunque

$$|\{E_k(m)\}_{k\in\mathcal{K}}| = |\mathcal{C}|$$

. Poichè $|\mathcal{K}| = |\mathcal{C}|$, segue che

$$|\{E_k(m)\}_{k\in\mathcal{K}}| = |\mathcal{K}|$$

Questo implica che non ci sono due chiavi distinte $k_1, k_2 \in \mathcal{K}$ tali che $E_{k_1}(m) = E_{k_2}(m)$. Dunque $\forall m$ e $\forall c$ esiste al massimo una chiave $k \in \mathcal{K}$ tale che $E_k(m) = c$.
($\Rightarrow$ 1.) Dimostriamo che $\forall k \in \mathcal{K}$, $P(K = k) = \frac{1}{|\mathcal{K}|}$.
Sia $n = |\mathcal{K}|$ e $\mathcal{M} = \{m_1, ..., m_n\}$ e fissiamo un testo cifrato c. Ordiniamo le chiavi $k_1, ..., k_n$ in modo tale che per ogni i valga $E_{k_i}(m_i) = c$. Per la segretezza perfetta abbiamo che per ogni i:

$$\begin{aligned} P(M = m_i) &= P(M = m_i|C = c) \\ &= \frac{P(C = c|M = m_i)P(M = m_i)}{P(C = c)} \\ &= \frac{P(K = k_i)P(M = m_i)}{P(C = c)} \end{aligned}$$

Ne segue che per ogni i, $P(K = k_i) = P(C = c)$, allora per ogni i, j:

$$P(K = k_i) = P(C = c) = P(K = k_j)$$

e dunque tutte le chiavi sono scelte con la stessa probabilità, che sarà uguale a $\frac{1}{|\mathcal{K}|}$ come voluto.
($\Leftarrow$) Dimostriamo ora che date le due condizioni il sistema è sicuro. Le condizioni 1. e 2. implicano che

$$P(C = c|M = m) = P(K = k) = \frac{1}{|\mathcal{K}|}$$

e questo vale per ogni distribuzione su $\mathcal{M}$. Dunque per ogni distribuzione su $\mathcal{M}$, ogni $m, m' \in \mathcal{M}$ e ogni $c \in \mathcal{C}$ vale

$$P(C = c|M = m) = \frac{1}{|\mathcal{K}|} = P(C = c|M = m')$$

□

Capitolo 2

Schemi computazionalmente sicuri

In questo capitolo andremo a definire la *sicurezza computazionale* e gli schemi che risultano sicuri secondo questa definizione. Tali schemi non soddisfano i requisiti della sicurezza perfetta, infatti possono essere violati avendo a disposizione abbastanza tempo, tuttavia anche usando il più veloce dei computer sarebbe richiesto un tempo molto più lungo della lunghezza media di una vita umana per portare a termine l'attacco. Ai fini pratici questo livello di sicurezza risulta sufficiente. Da qui in avanti utilizzeremo quindi l'approccio computazionale che è alla base della crittografia moderna.

2.1 L'approccio computazionale

Un cifrario deve essere praticamente, se non matematicamente, indecifrabile.

Questo è il primo dei sei principi enunciati da Kerckhoffs nel suo celebre articolo del 1883 *La cryptographie militaire*[5]. Esso riassume molto bene il concetto di sicurezza computazionale: non è necessario che uno schema sia perfettamente sicuro, ma è sufficiente che non venga violato in un tempo ragionevole e con una ragionevole probabilità di successo.

Abbiamo quindi due sostanziali differenze rispetto alla nozione di sicurezza perfetta:

1. La sicurezza viene preservata solamente contro avversari efficienti, che agiscono in un tempo polinomiale;

2. Gli avversari possono potenzialmente avere successo, ma con una bassissima probabilità.

Per spiegare cosa si intende con *avversari efficienti* e *bassissima probabilità* utilizzeremo l'approccio asintotico, che vede il tempo di esecuzione dell'avversario e la probabilità di successo come funzioni di un parametro $n \in \mathbb{Z}$, che chiameremo **parametro di sicurezza**[1]. Quando la parte onesta inizializza lo schema, ad esempio quando genera la chiave, sceglie un valore n che si suppone sia conosciuto anche alla parte avversa. Spesso questo valore viene inteso come lunghezza in bit della chiave. Il tempo di esecuzione dell'avversario e la sua probabilità di successo sono quindi entrambi funzioni di n.
Da qui in avanti indicheremo il parametro di sicurezza con 1^n, che indica nel codice binario una stringa di n bit uguali a 1.
Vediamo ora nel dettaglio cosa si intende con avversari efficienti e probabilità trascurabile.

2.1.1 L'avversario

Nella teoria computazionale l'avversario preso in considerazione è in possesso di risorse di calcolo limitate. Scegliamo quindi di rappresentarlo con un algoritmo $\mathcal{A}$ appartenente ad una particolare classe di complessità computazionale. In particolare, quando parliamo di *algoritmi efficienti*, intendiamo

[1] Un altro tipo di approccio è quello concreto, secondo il quale uno schema si dice (t, ε)-sicuro se viene violato nel tempo t con una probabilità minore di ε, dove t e ε sono due costanti positive, $\varepsilon \leq 1$. Potremmo dire che uno schema è sicuro se non viene violato in 200 anni con probabilità maggiore di 10^{-30}. Questo tipo di approccio risulta tuttavia molto incerto, e non dice nulla riguardo la probabilità di successo in un tempo minore o maggiore di quello indicato.

quegli algoritmi che terminano in un numero di passi polinomiale rispetto alla lunghezza dell'input, mentre consideriamo inefficienti quelli con complessità computazionale maggiore, ad esempio quelli che necessitano di un numero di operazioni esponenziale.

Definizione 2.1. Un algoritmo $\mathcal{A}$ si dice **polinomiale** se esiste un polinomio $p(n)$ tale che $\forall x \in \{0,1\}^*$, $\mathcal{A}(x)$ termina in un numero di passi $\leq p(|x|)$, dove $|x|$ indica la lunghezza della stringa x.

In altre parole, esistono due costanti a, c tali che, dato $n := |x|$, l'algoritmo termina in $a \cdot n^c$ passi.
Un altro fatto da considerare è che l'avversario può anche fare scelte casuali: ad esempio se la chiave è una stringa di n bit, può scoprirla con probabilità $\frac{1}{2^n}$ (ogni bit ha probabilità $\frac{1}{2}$ di essere scoperto). Questo valore, all'aumentare della lunghezza della chiave, tende a 0, ma per valori bassi di n la probabilità non è comunque nulla. Scegliamo quindi di rappresentare l'avversario tramite un algoritmo probabilistico. Un **algoritmo probabilistico** è un algoritmo che ha la capacità di "lanciare monete", nel senso che ha accesso ad una fonte di numeri casuali che all'occorrenza fornisce dei bit random, che sono indipendentemente uguali a 1 con probabilità $\frac{1}{2}$ o a 0 con probabilità $\frac{1}{2}$. Gli algoritmi probabilistici polinomiali in n vengono solitamente indicati con la sigla PPT (i.e. Probabilistic Polynomial-Time) e d'ora in avanti supporremo sempre che l'avversario sia di questo tipo. Non prendiamo in considerazione avversari che impieghino un tempo maggiore poichè immaginiamo che un avversario voglia essere efficiente e non voglia impiegare troppo tempo a violare lo schema, poichè in tal caso non costituirebbe un vero pericolo.

2.1.2 Probabilità trascurabile

Abbiamo accennato al fatto che nella crittografia moderna uno schema viene considerato sicuro se la probabilità che venga violato è molto bassa. Con *molto bassa* intendiamo una probabilità minore dell'inverso di qualsiasi polinomio in n, nel senso che per ogni costante c la probabilità di successo

dell'avversario è minore di n^{-c} per valori di n abbastanza alti. Una funzione che cresce più lentamente dell'inversa di ogni polinomio in n si dice trascurabile.

Definizione 2.2. Una funzione f si dice **trascurabile** se $\forall$ polinomio $p(n)$ $\exists$ N tale che $\forall n > N$ si ha $f(n) < \frac{1}{p(n)}$.

Le funzioni trascurabili sono utili perchè soddisfano alcune proprietà di chiusura. Vediamole nella seguente proposizione.

Proposizione 2.1.1. *Siano f_1 e f_2 due funzioni trascurabili. Allora*

1. *La funzione $f_3 := f_1 + f_2$ è trascurabile*
2. *$\forall$ polinomio p, la funzione $f_4 := p \cdot f_1$ è trascurabile*

La seconda parte della proposizione è particolarmente importante, perchè significa che se un evento ha probabilità trascurabile allora questa rimarrà trascurabile anche se l'esperimento viene ripetuto un numero polinomiale di volte. Gli eventi che avvengono con probabilità trascurabile possono essere ignorati ai fini pratici, così uno schema che viene violato con probabilità trascurabile risulta sicuro. Una definizione di sicurezza potrebbe quindi avere la seguente forma:

Uno schema è sicuro se tutti gli avversari del tipo PPT riescono a violare lo schema con probabilità trascurabile.

Anche in questo caso, la sicurezza è garantita per valori abbastanza alti del parametro di sicurezza n, anzi più alto è il valore, maggiore è la sicurezza. Nella maggior parte dei casi il parametro n è associato alla lunghezza della chiave, torniamo allora al concetto già visto in precedenza che più lunga è la chiave, maggiore è la sicurezza.

2.1.3 Dimostrazione per riduzione

Per dimostrare che uno schema è computazionalmente sicuro si rende necessario dimostrare la sua inviolabilità per ogni avversario che agisca in

tempo polinomiale. Una dimostrazione univoca per ogni tipo di avversario non è ancora stata trovata, perciò viene comunemente utilizzata la *tecnica per riduzione*. Questa consiste nel ridurre un problema che si suppone difficile, chiamiamolo B, al problema crittografico A da risolvere. Si ha allora che se si riesce a risolvere il problema A, ne segue che è risolto anche B.
Vogliamo dimostrare che lo schema crittografico A è sicuro; supponiamo che B sia un problema difficile da risolvere e supponiamo per assurdo che A sia facilmente violabile. Se riusciamo a ridurre il problema B a A, allora si ha che anche B è facilmente risolvibile, contraddicendo l'ipotesi.

2.2 Sicurezza computazionale

Riprendiamo gli algoritmi (Gen, E, D) definiti nel capitolo precedente, ma d'ora in avanti assumiamo che Gen prenda in input il parametro di sicurezza 1^n e restituisca una chiave casuale k che soddisfi $|k| \geq n$. Supponiamo inoltre che lo spazio dei messaggi sia l'insieme $\{0,1\}^*$ di tutte le stringhe binarie di lunghezza finita; se invece fissiamo una lunghezza massima $l(n)$, in tal caso lo schema è definito solo per messaggi $m \in \{0,1\}^{l(n)}$, e si dice a lunghezza fissata. Analizziamo, come nel capitolo precedente, il caso in cui un avversario abbia a disposizione un singolo testo cifrato, ma supponiamo che debba agire in un tempo polinomiale e definiamo la sicurezza computazionale in termini di **indistinguibilità**, cioè diciamo che uno schema è **computazionalmente sicuro** se possiede la proprietà dell'indistinguibilità. Questa definizione è basata su un esperimento, che chiameremo $PrivK^{eav}$, che coinvolge un avversario $\mathcal{A}$ e formalizza la sua incapacità di distinguere tra la cifratura di un testo in chiaro e quella di un altro.[2]
Dati uno schema $\Pi = (Gen, E, D)$, un avversario $\mathcal{A}$ e un valore n qualsiasi, definiamo l'esperimento $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ nel modo seguente:

[2] $PrivK^{eav}$ indica che stiamo considerando un cifrario a chiave simmetrica (Private-Key encryption) e un avversario con uno o più testi cifrati a disposizione (eavesdropping adversary).

1. Dato il parametro di sicurezza 1^n, l'avversario $\mathcal{A}$ sceglie due messaggi m_0, m_1 della stessa lunghezza.

2. Una chiave k è generata da $Gen(1^n)$ e viene scelto un bit $b \in \{0,1\}$ in maniera casuale.

3. Viene dato ad $\mathcal{A}$ un testo cifrato $c \leftarrow E_k(m_b)$.[3]

4. $\mathcal{A}$ sceglie un bit b'.

5. Il risultato dell'esperimento è 1 se $b' = b$, 0 altrimenti. Scriviamo $PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1$ se il risultato è 1 e in questo caso diciamo che $\mathcal{A}$ ha avuto successo.

In altre parole, $\mathcal{A}$ deve cercare di indovinare quale, tra i due messaggi m_0, m_1 è stato cifrato, e ha successo se la sua scelta è corretta. Osserviamo che $\mathcal{A}$ ha sempre la possibilità di avere successo con probabilità $\frac{1}{2}$ scegliendo b' in maniera casuale. La domanda è se sia possibile per l'avversario avere successo con probabilità maggiore di $\frac{1}{2}$.[4] Se Π è uno schema a lunghezza fissata $l(n)$, allora si richiede anche che $m_0, m_1 \in \{0,1\}^{l(n)}$.

Definizione 2.3. Uno schema crittografico a chiave simmetrica $\Pi = (Gen, E, D)$ soddisfa la proprietà di indistinguibilità se in presenza di un avversario $\mathcal{A}$[5] esiste una funzione trascurabile f tale che

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

Ribadiamo che l'avversario ha la possibilità di scegliere i due messaggi m_0 e m_1 ma, anche sapendo che c è la cifratura di uno dei due, non può

[3] Scriviamo $c \leftarrow E_k(m_b)$ per sottolineare che E_k può essere un algoritmo probabilistico che quindi restituisce diversi valori di c anche se viene usato con la stessa chiave k.

[4] Rimettendoci nelle condizioni del capitolo precedente, potremmo dire che uno schema è perfettamente sicuro se $\forall \mathcal{A}, \Pi$ si ha che $P[PrivK^{eav}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$ e questa definizione è l'analogo della Definizione 1.6.

[5] Si intende sempre un avversario probabilistico che abbia a disposizione solo il testo cifrato e che agisca in tempo polinomiale.

comunque determinare quale è stato cifrato.
Dimostriamo ora che *l'indistinguibilità implica che nessun bit di un messaggio può essere indovinato con probabilità maggiore di* $\frac{1}{2}$. Indichiamo con m^i l'i-esimo bit di m e poniamo $m^i = 0$ quando $i > |m|$.

Proposizione 2.2.1. *Sia* (Gen, E, D) *uno schema a chiave simmetrica con la proprietà di indistinguibilità. Allora per ogni avversario* $\mathcal{A}$ *e per ogni* i *esiste una funzione trascurabile* f *tale che*

$$P[\mathcal{A}(1^n, E_k(m)) = m^i] \leq \frac{1}{2} + f(n)$$

con m *scelto casualmente in* $\{0,1\}^n$.

Dimostrazione. Sia $\mathcal{A}$ un avversario probabilistico a tempo polinomiale e sia $\varepsilon(n) := P[\mathcal{A}(1^n, E_k(m)) = m^i] - \frac{1}{2}$. Sia $n \geq i$ e definiamo I_0^n l'insieme delle stringhe di lunghezza n che hanno l'i-esimo bit uguale a 0, I_1^n l'insieme delle stringhe di lunghezza n che hanno l'i-esimo bit uguale a 1. Ne segue che:

$$P[\mathcal{A}(1^n, E_k(m)) = m^i] = \frac{1}{2}P[\mathcal{A}(1^n, E_k(m_0)) = 0] + \frac{1}{2}P[\mathcal{A}(1^n, E_k(m_1)) = 1]$$

con m_0 scelto uniformemente da I_0^n e m_1 scelto uniformemente da I_1^n.
Consideriamo ora l'avversario $\mathcal{A}'$ così definito:

1. Dato l'input 1^n, sceglie $m_0 \leftarrow I_0^n$ e $m_1 \leftarrow I_1^n$ uniformemente.

2. Dopo aver ricevuto il testo cifrato c, lo dà in input ad $\mathcal{A}$. Restituisce $b' = 0$ se l'output di $\mathcal{A}$ è 0, $b' = 1$ se l'output di $\mathcal{A}$ è 1.

Poichè $\mathcal{A}$ è a tempo polinomiale, anche $\mathcal{A}'$ lo è.
Utilizzando la definizione dell'esperimento $PrivK^{eav}_{\mathcal{A}',\Pi}(n)$, si ha che $b' = b$ se e solo se $\mathcal{A}$ restituisce b dopo aver ricevuto $E_k(m_b)$. Dunque

$$\begin{aligned} P[PrivK^{eav}_{\mathcal{A}',\Pi}(n) = 1] &= P[\mathcal{A}(1^n, E_k(m_b)) = b] \\ &= \frac{1}{2}P[\mathcal{A}(1^n, E_k(m_0)) = 0] + \frac{1}{2}P[\mathcal{A}(1^n, E_k(m_1)) = 1] \\ &= P[\mathcal{A}(1^n, E_k(m)) = m^i] \\ &= \frac{1}{2} + \varepsilon(n) \end{aligned}$$

Assumendo che (Gen, E, D) abbia la proprietà di indistinguibilità, segue che $\varepsilon(n)$ è trascurabile. □

In definitiva, è chiaro che se l'avversario potesse indovinare l'i-esimo bit di m, potrebbe anche distinguere tra m_0 e m_1, scegliendoli in modo che abbiano l'i-esimo bit diverso.

2.2.1 Generatori di numeri pseudocasuali

Prima di passare alla costruzione di un sistema computazionalmente sicuro, dobbiamo introdurre la nozione di numero *pseudocasuale*. Una stringa pseudocasuale è una stringa che, agli occhi di un osservatore che agisce in tempo polinomiale, appare distribuita uniformemente (ossia come una stringa casuale). Consideriamo una distribuzione sulle stringhe di lunghezza l e diciamo che è pseudocasuale se è indistinguibile dalla distribuzione uniforme sulle stringhe di lunghezza l.
La pseudocasualità è utile nella costruzione di schemi a chiave privata perchè se un testo cifrato appare casuale allora non c'è modo per l'avversario di scoprire alcuna informazione riguardo il testo in chiaro. Il vantaggio di usare una stringa pseudocasuale invece di una casuale è che da una stringa casuale relativamente corta (ad esempio la chiave) è possibile generarne una più lunga pseudocasuale. Dunque una chiave corta può essere utilizzata per cifrare un messaggio molto più lungo, cosa che risulta impossibile quando viene richiesta la sicurezza perfetta.
Un *generatore pseudocasuale* è un algoritmo *deterministico* che prende in input una stringa casuale piuttosto corta e la trasforma in una stringa pseudocasuale più lunga. Nel far questo si richiede che ogni algoritmo PPT, in un esperimento simile a quello della definizione 2.4, scelga 1 (o 0) con la stessa probabilità dopo aver visto una stringa casuale o una pseudocasuale, ossia non sia in grado di distinguere, con probabilità maggiore di $\frac{1}{2}$, se la stringa ricevuta in input sia casuale o pseudocasuale. Nella definizione seguente

supponiamo che n sia la lunghezza dell'input casuale e $l(n)$ la lunghezza dell'output.

Definizione 2.4. Siano $l(n)$ un polinomio e G un algoritmo deterministico a tempo polinomiale tale che $\forall s \in \{0,1\}^n$, G restituisce una stringa di lunghezza $l(n)$. Allora G si dice un **generatore pseudocasuale** se valgono le seguenti condizioni:

1. (Espansione) $\forall n$ si ha che $l(n) > n$.
2. (Pseudocasualità) $\forall$ algoritmo PPT D, $\exists$ una funzione trascurabile f tale che
$$|P[D(r) = 1] - P[D(G(s)) = 1]| \leq f(n)$$
dove r è una stringa casuale $\in \{0,1\}^{l(n)}$, s è una stringa casuale $\in \{0,1\}^n$.

La funzione $l(n)$ è detta **fattore di espansione** di G.

Per un osservatore esponenziale una stringa pseudocasuale appare invece molto diversa da una casuale. Ad esempio[1], supponiamo che $l(n) = 2n$ e che quindi G raddoppi la lunghezza dell'input. La distribuzione uniforme su $\{0,1\}^{2n}$ è caratterizzata dal fatto che ognuna della 2^{2n} possibili stringhe è scelta con probabilità $\frac{1}{2^{2n}}$. Se consideriamo invece la distribuzione generata da G, si hanno solamente 2^n possibili stringhe, poichè G riceve un input di lunghezza n ed è deterministico. Dunque la probabilità che una stringa casuale di lunghezza $2n$ sia stata generata da G è $\frac{2^n}{2^{2n}} = \frac{1}{2^n}$. A questo punto, un osservatore D che abbia a disposizione un tempo illimitato, può scoprire se una stringa di lunghezza $2n$ è stata generata da G semplicemente controllando l'output $G(s)$ per ogni $s \in \{0,1\}^n$. Supponendo che D restituisca 1 se, data la stringa w, $\exists s \in \{0,1\}^n : G(s) = w$, si ha che:

$$|P[D(r) = 1] - P[D(G(s)) = 1]| = |2^{-n} - 1| = 1 - 2^{-n}$$

Questo valore è ovviamente molto alto, tuttavia gli avversari polinomiali non hanno il tempo di portare a termine tale procedura, a patto che n sia abbastanza grande.

2.3 Costruzione di uno schema crittografico sicuro

Alla luce di quanto detto precedentemente, possiamo ora costruire uno schema a lunghezza fissata che sia computazionalmente sicuro. Questo schema è molto simile al One-Time Pad, ad eccezione del fatto che trasformeremo la chiave in una stringa pseudocasuale più lunga da sommare al messaggio in chiaro tramite l'operazione XOR.
Sia G un generatore pseudocasuale con fattore di espansione l ($|G(s)| = l(|s|)$). Definiamo un sistema di cifratura Π a chiave privata per messaggi di lunghezza l nel modo seguente:

- **Gen**: dall'input 1^n, sceglie un $k \in \{0,1\}^n$ casualmente e lo restituisce come chiave.
- **E**: riceve in input una chiave $k \in \{0,1\}^n$ e un messaggio $m \in \{0,1\}^{l(n)}$ e restituisce il testo cifrato

$$c := G(k) \oplus m$$

- **D**: prende in input una chiave $k \in \{0,1\}^n$ e un testo cifrato $c \in \{0,1\}^{l(n)}$ e restituisce il messaggio in chiaro

$$m := G(k) \oplus c$$

Proveremo ora che questo schema ha la proprietà di indistinguibilità, assumendo che G sia un generatore pseudocasuale.

Teorema 2.3.1. *Se G è un generatore pseudocasuale, allora lo schema Π descritto sopra ha la proprietà di indistinguibilità.*

Dimostrazione. Sia $\mathcal{A}$ un avversario probabilistico a tempo polinomiale, e definiamo ε in questo modo:

$$\varepsilon(n) := P[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}$$

$\varepsilon(n)$ può essere considerato come il "vantaggio" di cui dispone $\mathcal{A}$. Utilizziamo $\mathcal{A}$ per costruire un algoritmo D che sappia distinguere le stringhe generate da G da quelle casuali, con probabilità $\varepsilon(n)$. Se $\mathcal{A}$ ha successo nell'esperimento $PrivK^{eav}$ allora D capisce se w è pseudocasuale, altrimenti no.
D prende in input una stringa $w \in \{0,1\}^{l(n)}$:

1. Dato 1^n, $\mathcal{A}$ sceglie due messaggi $m_0, m_1 \in \{0,1\}^{l(n)}$.
2. Scelto un bit random $b \leftarrow \{0,1\}$, viene dato ad $\mathcal{A}$ il testo cifrato $c := w \oplus m_b$.
3. Ottenuto l'output b' di $\mathcal{A}$, D restituisce 1 se $b' = b$, 0 altrimenti.

Definiamo poi uno schema $\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{E}, \widetilde{D})$ uguale al one-time pad ma che prenda in input il parametro di sicurezza 1^n, in modo che chiave e messaggio siano di lunghezza $l(n)$. Sappiamo che il one-time pad ha sicurezza perfetta, dunque si ha che

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

. Analizziamo il comportamento di D:

1. Se w viene scelto uniformemente dall'insieme $\{0,1\}^{l(n)}$, allora $\mathcal{A}$ si comporta come nell'esperimento $PrivK^{eav}_{\mathcal{A},\widetilde{\Pi}}(n)$ e dunque

$$P[D(w) = 1] = P[PrivK^{eav}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] = \frac{1}{2}$$

2. Se $w = G(k)$ per qualche $k \in \{0,1\}^n$ scelto casualmente, allora $\mathcal{A}$ si comporta come nell'esperimento $PrivK^{eav}_{\mathcal{A},\Pi}(n)$ e dunque

$$P[D(w) = 1] = P[D(G(k)) = 1] = P[PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \varepsilon(n)$$

.

Allora si ha che $|P[D(w) = 1] - P[D(G(k)) = 1]| = \varepsilon(n)$. Se assumiamo che G sia un generatore pseudocasuale allora ε deve essere trascurabile e questo implica che Π è computazionalmente sicuro. □

Bibliografia

[1] Jonathan Katz, Yehuda Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[2] W. Trappe, L.C. Washington, *Introduction to Cryptography with Coding Theory*, Pearson-Prentice Hall, 2006.

[3] Claude E. Shannon, *Communication Theory of Secrecy Systems*, Bell System Technical Journal, vol. 28-4, pp. 656-715, Oct. 1949.

[4] Claude E. Shannon, *A Mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, Oct. 1948.

[5] Auguste Kerckhoffs, *La cryptographie militaire*, Journal des sciences militaires, vol. IX, pp. 5-38, Jan. 1883, pp. 161-191, Février 1883.