Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

SICUREZZA DI SHANNON E SICUREZZA COMPUTAZIONALE

Tesi di Laurea in Crittografia

Relatore:
Chiar.mo Prof.
DAVIDE ALIFFI

Presentata da:
PIERPAOLO COLAGÈ

III Sessione
A. A. 2015/2016

Alla mia grande famiglia.

Indice

Introduzione **7**

1 Sicurezza dimostrabile di Shannon **9**
1.1 Richiami di probabilità 9
1.2 Schema crittografico a chiave privata 11
1.3 Sicurezza di Shannon e sicurezza perfetta 14
1.4 Teorema di Shannon 20

2 Sicurezza computazionale **27**
2.1 Efficienza di calcolo e avversario efficiente 27
2.2 Funzioni unidirezionali 31
2.3 Indistinguibilità e pseudo-casualità 32
2.4 Definizione computazionale di sicurezza 35

Conclusione **37**

Bibliografia **39**

Introduzione

La parola crittografia deriva dalle parole greche *kryptòs* che significa "nascosto" e *graphìa* che significa "scrittura". La crittografia quindi si occupa delle "scritture nascoste", ossia cerca di creare dei metodi per rendere un messaggio incomprensibile a persone non autorizzate a leggerlo. Si costruiscono quindi degli schemi crittografici per trasformare un messaggio in un testo cifrato in modo che due parti oneste, Alice e Bob, possono scambiarselo senza che un intercettatore, Eve, possa leggerlo. Viene quindi naturale pensare che tali schemi di crittografia devono essere sicuri di fronte a dei possibili avversari. In questo elaborato andremo ad analizzare la sicurezza dei sistemi crittografici a chiave privata, sistemi crittografici in cui l'unica parte che deve rimanere segreta è appunto la chiave. Distingueremo due tipi di sicurezza:

- *sicurezza di Shannon*;
- *sicurezza computazionale*.

La sicurezza di Shannon suppone che gli avversari abbiano capacità computazionale illimitata, mentre nel secondo tipo di sicurezza si considerano avversari con potenza di calcolo limitata, perchè, nella pratica, sono queste le reali minacce agli schemi crittografici.

Nel primo capitolo, dopo aver dato delle nozioni di base di probabilità, daremo due definizioni di sicurezza rispetto ad avversari computazionalmente illimitati, quella di Shannon e la sicurezza perfetta, che dimostreremo essere equivalenti. Vedremo inoltre, tramite dei teoremi, alcune proprietà che caratterizzano gli schemi crittografici a chiave privata sicuri secondo Shannon. Queste proprietà ci porteranno ad affermare che i sistemi crittografici sicuri secondo Shannon presentano dei problemi che non consentono l'attuazione pratica di tali schemi. Descriveremo il cifrario di Vernam e dopo averlo analizzato dimostreremo che è sicuro secondo Shannon ma è inutilizzabile in pratica. Infine costruiremo un semplice schema crittografico e dimostreremo in più modi, tramite la teoria studiata precedentemente, che esso non ha sicurezza di Shannon.

Nel secondo capitolo la nostra attenzione si sposterà sulla sicurezza computazionale, perchè è questa che nella pratica garantisce lo scambio di messaggi in modo sicuro. Prima di dare la definizione di sicurezza computazionale definiremo il concetto di efficienza di calcolo e di conseguenza cosa intendiamo con avversario efficiente con capacità computazionale limitata. Rappresenteremo gli avversari tramite algoritmi, pertanto, nella prima parte di questo capitolo, daremo delle definizioni che ci permetteranno di capire cosa sia un

algoritmo e in particolare cosa sia un algoritmo efficiente. Successivamente daremo le definizioni di *funzioni unidirezionali*, *indistinguibilità computazionale* e *generatori pseudo-casuali*, strumenti necessari per la costruzione di schemi crittografici efficienti e computazionalmente sicuri. Analizzeremo due importanti proprietà dell'indistinguibilità computazionale, in particolare la transitività tramite il lemma ibrido. Dopo aver dato la definizione di schema crittografico *single-message secure*, cioè uno schema crittografico sicuro computazionalmente solo per la cifratura di un singolo messaggio, descriveremo un sistema crittografico efficiente e dimostreremo che questo è *single-message secure*.

Capitolo 1

Sicurezza dimostrabile di Shannon

1.1 Richiami di probabilità

Introduciamo delle nozioni di probabilità che saranno lo strumento di base per lo studio dei sistemi crittografici e della loro sicurezza.

Consideriamo uno *spazio di probabilità discreto* (Ω, Pr), dove:

- Ω è un insieme non vuoto, finito o numerabile;
- Pr è una *probabilità* ossia una funzione

$$Pr : \mathcal{P}(\Omega) \longrightarrow [0, 1]$$

tale che $Pr[\Omega] = 1$ ed è σ-additiva (con $\mathcal{P}(\Omega)$ indichiamo l'insieme delle parti di Ω).

Sia un evento $A \in \mathcal{P}(\Omega)$, con $Pr[A]$ denotiamo semplicemente la probabilità che si verifichi tale evento. Inoltre, diremo che A è un evento *non trascurabile* se $Pr[A] > 0$.

Diamo ora la definizione di *probabilità condizionata* che ci permetterà di analizzare come il verificarsi di un evento B influenzi la probabilità di un altro evento A.

Definizione 1.1. Siano A e B due eventi con B non trascurabile, allora:

$$Pr[A \mid B] := \frac{Pr[A \cap B]}{Pr[B]}, \qquad A, B \in \mathcal{P}(\Omega) \tag{1.1}$$

è detta *probabilità di A condizionata a B*.

Proposizione 1.1.1 (Formula della probabilità totale)**.** *Sia $(B_i)_{i\in I}$ una partizione finita o numerabile di Ω, con B_i eventi non trascurabili per ogni $i \in I$, allora vale:*

$$Pr[A] = \sum_{i\in I} Pr[A \mid B_i]Pr[B_i], \qquad A \in \mathcal{P}(\Omega).$$

Dimostrazione. Poichè,

$$A = A \cap \Omega = A \cap \left(\bigcup_{i\in I} B_i\right) = \bigcup_{i\in I}(A \cap B_i)$$

dove l'unione è disgiunta, per la σ -additività di Pr si ha:

$$Pr[A] = \sum_{i\in I} Pr[A \cap B_i] = \sum_{i\in I} Pr[A \mid B_i]Pr[B_i]$$

□

Vediamo ora un teorema che mette in relazione $Pr[A \mid B]$ con $Pr[B \mid A]$.

Teorema 1.1.2 (Formula di Bayes)**.** *Siano A e B due eventi non trascurabili, vale:*

$$Pr[B \mid A] = \frac{Pr[A \mid B]Pr[B]}{Pr[A]}. \tag{1.2}$$

Dimostrazione. La formula (1.2) equivale a:

$$Pr[B \mid A]Pr[A] = Pr[A \mid B]Pr[B]$$

che per la definizione 1.1 è come dire:

$$Pr[B \cap A] = Pr[A \cap B]$$

che risulta essere ovviamente vera. □

Dalla definizione di probabilità condizionata abbiamo, in generale, che:

$$Pr[A \cap B] = Pr[A \mid B]Pr[B],$$

quindi i due eventi A e B si dicono *indipendenti* se

$$Pr[A \cap B] = Pr[A]Pr[B].$$

1.2 Schema crittografico a chiave privata

Consideriamo il seguente problema: due parti, Alice e Bob, vogliono scambiarsi messaggi in un canale insicuro (cioè un canale "aperto"). In particolare Alice vuole inviare messaggi, chiamati *testi in chiaro*, a Bob sul canale insicuro mantenendo la segretezza del messaggio anche di fronte a un avversario, Eve, che ascolta tutti i messaggi sul canale. Per iniziare la loro comunicazione Alice e Bob dovranno accordarsi su un "codice segreto" che successivamente useranno per lo scambio dei messaggi. Il codice deve essere composto da una chiave, un algoritmo Enc per cifrare messaggi di testo in chiaro in *testo cifrato* e un algoritmo Dec per trasformare i testi cifrati in testi in chiaro. Entrambi gli algoritmi richiedono l'uso della chiave per svolgere i loro compiti. Quindi Alice usa la chiave per creare il testo cifrato, invia il messaggio a Bob che a sua volta, sempre tramite la chiave, riesce a decifrarlo, trasformandolo nuovamente nel testo in chiaro. Per formalizzare quest'ultima operazione abbiamo bisogno di un ulteriore algoritmo, Gen, che genera la chiave k che Alice e Bob useranno per cifrare e decifrare i messaggi.

Osservazione 1 (Principio di Kerckhoffs). La sicurezza di un crittosistema non deve dipendere dal tener celato il crittosistema stesso, ma deve dipendere solo dal tener nascosta la chiave. Quindi gli algoritmi Gen, Enc e Dec possono essere resi pubblici, mentre l'unica informazione che deve essere segreta è la chiave k.

Oggi il principio di Kerckhoffs viene inteso nel senso che gli algoritmi devono essere completamente pubblici. In questo modo gli algoritmi possono essere studiati dalla comunità dei crittografi e se presentano delle debolezze queste verranno scoperte più facilmente.

Definizione 1.2 (Schema crittografico a chiave privata)**.** La terna (Gen, Enc, Dec) si dice uno *schema crittografico a chiave privata* sullo spazio dei messaggi $\mathcal{M}$ e sullo spazio delle chiavi $\mathcal{K}$ se vale quanto segue:

1. Gen (algoritmo di generazioni delle chiavi) è un algoritmo probabilistico che restituisce una chiave k tale che $k \in \mathcal{K}$. Indichiamo con $k \leftarrow Gen$ il processo di generazione di una chiave k;

2. Enc (algoritmo di cifratura) è un algoritmo, potenzialmente probabilistico, che prende in input una chiave $k \in \mathcal{K}$ e un messaggio $m \in \mathcal{M}$ e restituisce un testo cifrato c. Denotiamo con $\mathcal{C}$ l'insieme di tutti i possibili testi cifrati che possono essere emessi da $Enc_k(m)$ per ogni possibile scelta di $k \in \mathcal{K}$ e $m \in \mathcal{M}$. Indichiamo con $c \leftarrow Enc_k(m)$ l'operazione di cifratura del messaggio m tramite l'algoritmo Enc con chiave k;

3. *Dec* (algoritmo di decifrazione) è un algoritmo deterministico che prende in input un chiave $k \in \mathcal{K}$, un testo cifrato $c \in \mathcal{C}$ restituendo il testo in chiaro $m \in \mathcal{M}$. Indichiamo con $m \leftarrow Dec_k(c)$ l'operazione di decifrazione del testo cifrato c tramite l'algoritmo Dec con chiave k;

4. per ogni $m \in \mathcal{M}$ e per ogni chiave data da $k \leftarrow Gen$,

$$Pr[Dec_k(Enc_k(m)) = m] = 1.$$

Per semplificare la notazione diremo che $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ è uno schema crittografico a chiave privata se (Gen, Enc, Dec) è uno schema crittografico a chiave privata sullo spazio dei messaggi $\mathcal{M}$ e sullo spazio delle chiavi $\mathcal{K}$.

Osservazione 2. Nella definizione di schema di crittografia a chiave privata abbiamo detto che l'algoritmo Dec è un algoritmo deterministico, cioè per ogni $k \in \mathcal{K}$, per ogni $m \in \mathcal{M}$ e ogni $c \in \mathcal{C}$ generato da $Enc_k(m)$, si ha che $Dec_k(c) = m$; 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 più precisamente, otterrebbe un testo in chiaro che potrebbe non corrispondere al testo scelto da Alice.

I primi sistemi crittografici a chiave privata funzionavano in modo monoalfabetico, vale a dire, utilizzavano un alfabeto per il testo in chiaro e una permutazione dello stesso per il testo cifrato. La permutazione utilizzata costituiva la chiave del sistema. Quindi durante la cifratura, ad ogni lettera del testo in chiaro viene associata la corrispondente lettera dell'alfabeto permutato. Utilizzando la permutazione inversa dal testo cifrato si ricava il testo in chiaro.

Esempio 1.1 (Cifrario di Cesare)**.** Il *cifrario di Cesare* è uno dei più antichi algoritmi crittografici di cui si abbia traccia storica. È un *cifrario a sostituzione monoalfabetica* in cui ogni lettera del testo in chiaro è sostituita nel testo cifrato dalla lettera che si trova un certo numero di posizioni dopo nell'alfabeto. In particolare Cesare usava uno spostamento di tre posizioni (in questo caso la chiave è la traslazione di 3 posizioni), quindi a un messaggio del tipo:

ATTACCAREIGALLIALLAORASESTA

corrispondeva il testo cifrato:

DWWDFFDUHLJDOOLDOODRUDVHVWD.

Per decifrare il testo cifrato in questo caso basta sostituire ogni lettera con la lettera dell'alfabeto tre posizioni indietro.

Questo tipo di cifratura viene facilmente forzata dalla probabilità della distribuzione delle lettere nella lingua usata per il messaggio, nel senso che il carattere più frequente del testo cifrato corrisponderà molto probabilmente alla lettera più usata nella lingua del messaggio.

Successivamente questo metodo di cifratura venne sostituito da quello polialfabetico. I *cifrari a sostituzione polialfabetici* fanno uso di un numero più o meno grande di "alfabeti" usando un determinato ordine, che costituisce la chiave. Questo tipo di cifrario appiattisce la frequenza delle lettere e quindi, a differenza dei cifrari monoalfabetici, è resistente ad attacchi basati sulla distribuzione delle lettere. Nella seguente definizione descriviamo il *cifrario di Vigenère*, il più semplice dei cifrari a sostituzione polialfabetica.

Definizione 1.3 (Cifrario di Vigenère)**.** Sia n un intero positivo, definiamo $\mathcal{M} = \mathcal{K} = \mathcal{C} = (\mathbb{Z}_{26})^n$, inoltre consideriamo la chiave $k = (k_1, k_2, \ldots, k_n)$, quindi definiamo gli algoritmi di cifratura e decifrazione come:

$$Enc_k((x_1, x_2, \ldots, x_n)) = (x_1 + k_1, x_2 + k_2, \ldots, x_n + k_n)$$

e

$$Dec_k((y_1, y_2, \ldots, y_n)) = (y_1 - k_1, y_2 - k_2, \ldots, y_n - k_n)$$

dove ogni operazione è eseguita in $\mathbb{Z}_{26}$.

Nell'ultima definizione è stato considerato l'alfabeto inglese ed è stata usata la corrispondenza A $\leftrightarrow$ 0, B $\leftrightarrow$ 1, ..., Z $\leftrightarrow$ 25.

Diamo un piccolo esempio.

Esempio 1.2. Supponiamo $n = 10$ e che la chiave sia:

ACCADEMICO,

che corrisponde alla sequenza numerica:

$$(0, 2, 2, 0, 3, 4, 12, 8, 2, 14) = k.$$

Vogliamo cifrare il messaggio:

MATEMATICA

che invece corrisponde a:

$$(12, 0, 19, 4, 12, 0, 19, 8, 2, 0),$$

quindi:

$$Enc_{\mathrm{k}}((12, 0, 19, 4, 12, 0, 19, 8, 2, 0)) =$$

$$(12+0, 0+2, 19+2, 4+0, 12+3, 0+4, 19+12, 8+8, 2+2, 0+14)$$

e ricordando che la somma è fatta in $\mathbb{Z}_{26}$, otteniamo:

$$(12, 2, 21, 4, 15, 4, 5, 16, 4, 14)$$

che corrisponde al testo cifrato

MCVEPEFQEO.

Osserviamo che le due M della parola MATEMATICA (come le tre A e le due T) vengono cifrate in modo diverso tramite il cifrario di Vigenère, cosa che non sarebbe successa con un cifrario monoalfabetico. Un'ulteriore differenza tra i due tipi di cifrari è che nel cifrario del primo esempio, le possibili chiavi sono 26, mentre nei cifrari polialfabetici le chiavi possibili sono 26^n e per esempio con $n = 5$ otteniamo un numero di chiavi dell'ordine di 10^7. Questo è già abbastanza grande da impedire la ricerca esaustiva delle chiavi a mano (ma non con un computer), a differenza del caso monoalfabetico in cui le 26 chiavi si possono provare, a mano, in tempi brevi.[1]

1.3 Sicurezza di Shannon e sicurezza perfetta

Nelle definizioni e nei teoremi che seguiranno ci riferiremo a distribuzioni di probabilità su $\mathcal{K}$, $\mathcal{M}$ e $\mathcal{C}$. La distribuzione su $\mathcal{K}$ è semplicemente quella che viene definita eseguendo l'algoritmo Gen. Quindi, senza perdere di generalità, possiamo assumere che Gen scelga la chiave uniformemente in $\mathcal{K}$. Per $k \in \mathcal{K}$ indichiamo con $Pr[K = k]$ la probabilità che la chiave generata da Gen sia uguale a k (formalmente, K è una variabile aleatoria che denota il valore della chiave). Analogamente, per $m \in \mathcal{M}$ denotiamo con $Pr[M = m]$ la probabilità che il messaggio sia uguale a m. Il fatto che il messaggio viene scelto secondo una distribuzione (invece di essere fissato) serve per indicare che, almeno dal punto di vista dell'avversario, messaggi diversi hanno diverse probabilità di essere inviati. Le distribuzioni su $\mathcal{K}$ e $\mathcal{M}$ sono indipendenti, cioè la chiave e il messaggio sono scelti in modo indipendente; infatti la chiave è scelta e fissata prima di conoscere i messaggi. Infine, per $c \in \mathcal{C}$ scriviamo $Pr[C = c]$ per indicare la probabilità che il testo cifrato sia c. Dato l'algoritmo Enc, la distribuzione su $\mathcal{C}$ è completamente determinata dalle distribuzioni su $\mathcal{K}$ e $\mathcal{M}$.

Iniziamo con una definizione intuitiva di sicurezza: *date alcune informazioni a priori, l'avversario non può ottenere ulteriori informazioni sul testo*

[1] Nel cifrario monoalfabetico più generale le chiavi sono (26)!.

in chiaro osservando il testo cifrato. Tale nozione di sicurezza venne formalizzata da Claude Shannon nel 1949, dando il via allo studio moderno della crittografia.

Nel seguito considereremo solo distribuzioni su $\mathcal{M}$ e su $\mathcal{C}$ che non diano probabilità zero, per ogni $m \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$.

Definizione 1.4 (Sicurezza di Shannon)**.** Uno schema crittografico a chiave privata $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ si dice *sicuro secondo Shannon rispetto alla distribuzione D sullo spazio dei messaggi $\mathcal{M}$* se per ogni $m \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$, vale:

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

Si dirà che uno schema è *sicuro secondo Shannon* se è sicuro secondo Shannon rispetto a ogni distribuzione D sullo spazio dei messaggi $\mathcal{M}$.

La probabilità è presa rispetto a una scelta casuale della chiave k tramite l'algoritmo Gen. Inoltre, la quantità a sinistra nella precedente espressione rappresenta l'avversario ed è una distribuzione a posteriori sui testi in chiaro, dopo aver osservato un testo cifrato; la quantità a destra, invece, rappresenta la distribuzione a priori. Dal momento che queste distribuzioni devono essere uguali, questa definizione richiede che l'avversario non guadagni ulteriori informazioni osservando il testo cifrato.

Nel prossimo lemma vediamo una formulazione equivalente della definizione 1.4.

Lemma 1.3.1. *Uno schema crittografico $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ è sicuro secondo Shannon se e solo se per ogni distribuzione su $\mathcal{M}$, $\forall m \in \mathcal{M}$ e $\forall c \in \mathcal{C}$, vale:*

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

Dimostrazione. Fissiamo una distribuzione su $\mathcal{M}$, un $m \in \mathcal{M}$ e un $c \in \mathcal{C}$. Supponiamo che $Pr[C = c \mid M = m] = Pr[C = c]$, allora moltiplicando ambo i membri per $Pr[M = m]/Pr[C = c]$ otteniamo:

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

e per il teorema 1.1.2 il lato sinistro dell'equazione è uguale a:

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

ottenendo così:

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

che garantisce la sicurezza di Shannon del sistema. Viceversa, supponiamo che lo schema crittografico sia sicuro secondo Shannon, cioè che valga:

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

moltiplicando sia la parte sinistra che la parte destra dell'equazione precedente per $Pr[C = c]/Pr[M = m]$, otteniamo:

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

e sempre per il teorema 1.1.2 il lato sinistro dell'equazione è uguale a:

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

ottenendo così:

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

cioè proprio la formula dell'enunciato. □

Diamo ora un'altra definizione di sicurezza per sistemi crittografici. La nozione di *sicurezza perfetta* richiede che la distribuzione dei testi cifrati per qualsiasi coppia di messaggi sia identica; vale a dire, la distribuzione di probabilità su $\mathcal{C}$ è indipendente dal testo in chiaro. Anche questa definizione ribadisce che i testi cifrati non consentono ad un eventuale avversario di ricavare alcuna informazione circa il testo in chiaro.

Definizione 1.5 (Sicurezza perfetta)**.** $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ si dice uno schema crittografico a chiave privata *perfettamente sicuro* se per ogni distribuzione di probabilità su $\mathcal{M}$, per ogni $m_1, m_2 \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$ si ha:

$$Pr[C = c \mid M = m_1] = Pr[C = c \mid M = m_2].$$

Nel seguente teorema dimostriamo l'equivalenza tra le due definizioni di sicurezza per schemi crittografici date in precedenza.

Teorema 1.3.2. *Uno schema crittografico a chiave privata è sicuro secondo Shannon se e solo se è perfettamente sicuro.*

Dimostrazione. **Sicurezza di Shannon ⇒ Sicurezza perfetta**. Assumiamo che lo schema crittografico sia sicuro secondo Shannon, fissiamo i messaggi $m_1, m_2 \in \mathcal{M}$ e un testo cifrato $c \in \mathcal{C}$; valgono le uguaglianze:

1. $Pr[M = m_1 \mid C = c] = Pr[M = m_1]$;

2. $Pr[M = m_2 \mid C = c] = Pr[M = m_2]$.

Per il lemma 1.3.1 otteniamo che:

$$Pr[C = c \mid M = m_1] = Pr[C = c] = Pr[C = c \mid M = m_2]$$

e questo conclude la prima parte della dimostrazione.
Sicurezza perfetta ⇒ Sicurezza di Shannon. Supponiamo ora che il crittosistema sia perfettamente sicuro, cioè per ogni distribuzione su $\mathcal{M}$, per ogni m_1, $m_2 \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$, vale:

$$Pr[C = c \mid M = m_1] = Pr[C = c \mid M = m_2].$$

Fissiamo ora una distribuzione su $\mathcal{M}$, un arbitrario $m_1 \in \mathcal{M}$ e un $c \in \mathcal{C}$, definiamo:

$$p := Pr[C = c \mid M = m_1],$$

da $Pr[C = c \mid M = m] = Pr[C = c \mid M = m_1]$ per ogni $m \in \mathcal{M}$, abbiamo:

$$\begin{aligned} Pr[C = c] &= \sum_{m \in \mathcal{M}} Pr[C = c \mid M = m] Pr[M = m] \\ &= \sum_{m \in \mathcal{M}} p \cdot (Pr[M = m]) \\ &= p \cdot \sum_{m \in \mathcal{M}} Pr[M = m] \\ &= p \\ &= Pr[C = c \mid M = m_1] \end{aligned}$$

dove la prima uguaglianza deriva dalla proposizione 1.1.1.
Poichè m_1 era arbitrario, abbiamo mostrato che:

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

per ogni $c \in \mathcal{C}$ e per ogni $m \in \mathcal{M}$. Applicando il lemma 1.3.1 concludiamo che lo schema crittografico è sicuro secondo Shannon. □

Descriviamo ora il *cifrario di Vernam*. Questo è un sistema crittografico basato sul cifrario di Vigenère (definizione 1.3), al quale si aggiunge il requisito che la chiave di cifratura sia lunga quanto il testo e che non sia riutilizzabile. Per quest'ultima proprietà il cifrario di Vernam viene spesso chiamato *OTP*, acronimo per l'inglese One Time Pad, che letteralmente significa "blocco monouso".

Prima di dare la definizione formale del cifrario di Vernam, introduciamo il concetto di *XOR* (exlusive-or). Indichiamo l'operazione di XOR con il

simbolo $\oplus$, il quale è un operatore logico che si applica a due variabili e restituisce VERO se e solo se le variabili sono diverse (la prima VERO e la seconda FALSO o viceversa), altrimenti restituisce FALSO (se le due variabili sono entrambe VERO o entrambe FALSO). In particolare noi useremo questo operatore in $\mathbb{Z}_2$ (in questo caso con il simbolo $\oplus$ stiamo indicando la *somma modulo 2*) e quindi avremo i seguenti possibili casi: $0 \oplus 0 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$ e $1 \oplus 1 = 0$. Inoltre se abbiamo a e $b \in (\mathbb{Z}_2)^l$, con l intero maggiore di zero, $a \oplus b$ significa:

$$a \oplus b = (a_1, a_2, \ldots, a_l) \oplus (b_1, b_2, \ldots, b_l) = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_l \oplus b_l),$$

dove $a_i, b_j \in \mathbb{Z}_2$ $\forall i, j = 1, \ldots, l$.

Definizione 1.6 (Cifrario di Vernam)**.** Sia l un intero positivo, definiamo $\mathcal{M} = \mathcal{K} = \mathcal{C} = (\mathbb{Z}_2)^l$. L'algoritmo di generazione delle chiavi, Gen, lavora scegliendo una stringa in $(\mathbb{Z}_2)^l$ seguendo la distribuzione uniforme (ognuna delle 2^l stringhe in $\mathcal{K}$ è scelta come chiave con una probabilità uguale a 2^{-l}). Inoltre data una chiave $k \in \mathcal{K}$ e un testo in chiaro $m \in \mathcal{M}$, si definisce l'algoritmo di cifratura come:

$$Enc_k(m) = k \oplus m = c, \qquad c \in \mathcal{C}.$$

Dato un testo cifrato $c \in \mathcal{C}$ si definisce l'algoritmo di decifrazione con chiave $k \in \mathcal{K}$ nel seguente modo:

$$Dec_k(c) = k \oplus c = m, \qquad m \in \mathcal{M}.$$

Osservazione 3. Notiamo che per ogni chiave $k \in \mathcal{K}$ e per ogni testo in chiaro $m \in \mathcal{M}$, si ha:

$$Dec_k(Enc_k(m)) = k \oplus k \oplus m = m$$

e questo ci dice che il cifrario è ben definito.

Osservazione 4. Per cifrare un messaggio con il cifrario di Vernam si deve prima "tradurre" il messaggio in codice binario, cioè una stringa in cui ogni elemento, chiamato *bit*, può assumere i valori 0 oppure 1. Storicamente questa operazione veniva eseguita tramite il *codice di Baudot* che faceva corrispondere ad ognuna delle 26 lettere dell'alfabeto inglese una stringa di 5 bit (con 5 bit si hanno $2^5 = 32$ combinazioni, le 6 combinazioni in eccesso corrispondono a dei codici di controllo). Una volta fatta questa corrispondenza il messaggio appartiene effettivamente a $\mathcal{M}$, e si può quindi passare alla cifratura del testo in chiaro.

Mostriamo un esempio per comprendere meglio il funzionamento del cifrario di Vernam.

Esempio 1.3. Consideriamo la chiave **ZBU** che, tramite il codice di Baudot, corrisponde alla sequenza:

$$(\underbrace{1,0,0,0,1}_{\mathbf{Z}},\underbrace{1,1,0,0,1}_{\mathbf{B}},\underbrace{0,0,1,1,1}_{\mathbf{U}}) = k$$

e vogliamo cifrare il messaggio **PER** che invece corrisponde a:

$$(\underbrace{1,0,1,1,0}_{\mathbf{P}},\underbrace{0,0,0,0,1}_{\mathbf{E}},\underbrace{0,1,0,1,0}_{\mathbf{R}}) = m,$$

allora:

$$Enc_k(m) = k \oplus m =$$
$$(\underbrace{0,0,1,1,1}_{\mathbf{U}},\underbrace{1,1,0,0,0}_{\mathbf{O}},\underbrace{0,1,1,0,1}_{\mathbf{F}}) = c$$

che corrisponde al testo cifrato **UOF**.

Il cifrario di Vernam, brevettato nel 1917 da Gilbert S. Vernam, è l'unico sistema crittografico la cui sicurezza sia comprovata da una dimostrazione matematica e per questo si è guadagnato il titolo di "cifrario perfetto". La prima dimostrazione della sua inviolabilità fu pubblicata nel 1949 da Claude Shannon nel suo articolo *Communication Theory of Secrecy Systems*. Abbiamo quindi il seguente teorema:

Teorema 1.3.3. *Il cifrario di Vernam è perfettamente sicuro.*

Dimostrazione. Fissiamo una distribuzione su $\mathcal{M}$, consideriamo un arbitrario $m \in \mathcal{M}$ e un arbitrario $c \in \mathcal{C}$. L'osservazione fondamentale è che per il cifrario di Vernam vale:

$$Pr[C = c \mid M = m] = Pr[M \oplus K = c \mid M = m] =$$
$$Pr[m \oplus K = c] = Pr[K = m \oplus c] = \frac{1}{2^l}.$$

Poichè questo vale per tutte le distribuzioni e per tutti gli m, si ha che per ogni distribuzioni di probabilità su $\mathcal{M}$, per ogni $m_1, m_2 \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$, vale:

$$Pr[C = c \mid M = m_1] = \frac{1}{2^l} = Pr[C = c \mid M = m_2],$$

e questo dimostra che il cifrario di Vernam è perfettamente sicuro. □

Osservazione 5. Per il teorema 1.3.2 sappiamo che un cifrario è sicuro secondo Shannon se e solo se è perfettamente sicuro, quindi dal teorema precedente sappiamo che il cifrario di Vernam è sicuro anche secondo Shannon.

Abbiamo appena visto che il cifrario di Vernam è sicuro secondo Shannon. Tuttavia, purtroppo, questo schema crittografico presenta alcuni problemi pratici. Il più importante è che la chiave deve essere lunga quanto il messaggio e questo significa che una chiave lunga deve essere conservata in modo sicuro, cosa altamente problematica in pratica e spesso non realizzabile. Quindi abbiamo un "limite" sulla lunghezza dei messaggi: più questi sono lunghi e più sarà difficile memorizzare, in modo sicuro, la chiave. Inoltre, non potendo scambiare chiavi di lunghezza infinita, si dovrebbe avere un limite superiore per la lunghezza dei messaggi, altrimenti si potrebbe dover cifrare un messaggio che sia più lungo della chiave. Nel caso di una chiave più corta del messaggio si potrebbe pensare di cifrare il testo in chiaro concatenando la stessa chiave più volte in modo da coprire tutta la lunghezza del messaggio. Fare ciò è come usare la medesima chiave per cifrare più di un messaggio, e questo compromette la sicurezza del cifrario. Infatti, consideriamo due messaggi m e m' e supponiamo che questi siano cifrati con la stessa chiave k. Se un avversario intercetta i due testi cifrati $c = m \oplus k$ e $c' = m' \oplus k$, allora, può calcolare:

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

e quindi imparare qualcosa sui testi in chiaro.

1.4 Teorema di Shannon

In questa sezione daremo altre condizioni per cui uno schema crittografico si possa considerare sicuro secondo Shannon. Mostreremo inoltre dei limiti pratici dei cifrari con sicurezza di Shannon, dando conferma dei problemi del cifrario di Vernam discussi nel paragrafo precedente.

In particolare, dimostreremo che un qualsiasi sistema crittografico che ha sicurezza di Shannon deve avere lo spazio delle chiavi grande almeno quanto lo spazio dei messaggi. Se questi due spazi contengono rispettivamente chiavi e testi in chiaro a lunghezza fissata, allora abbiamo che la chiave deve essere lunga almeno quanto il messaggio. Quindi il problema della lunghezza della chiave di cifratura non è inerente solo al cifrario di Vernam, ma in generale riguarda un qualsiasi sistema crittografico sicuro secondo Shannon. Abbiamo quindi il seguente teorema:

Teorema 1.4.1. *Se uno schema crittografico* $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ *è sicuro secondo Shannon, allora* $|\mathcal{K}| \geq |\mathcal{M}|$.

Dimostrazione. Mostreremo che se $|\mathcal{K}| < |\mathcal{M}|$ allora lo schema crittografico non è sicuro secondo Shannon. Quindi, supponiamo che $|\mathcal{K}| < |\mathcal{M}|$ e consideriamo una distribuzione su $\mathcal{M}$. Sia $c \in \mathcal{C}$ un testo cifrato che si verifica con

probabilità diversa da zero, indichiamo con $\mathcal{M}(c)$ l'insieme di tutti i messaggi che sono possibili decodifiche per c, cioè:

$$\mathcal{M}(c) := \left\{ \widehat{m} \mid \widehat{m} = Dec_{\widehat{k}}(c), \ per \ qualche \ \widehat{k} \in \mathcal{K} \right\}.$$

Chiaramente, $|\mathcal{M}(c)| \leq |\mathcal{K}|$ poichè per ogni messaggio $\widehat{m} \in \mathcal{M}(c)$ possiamo trovare almeno una chiave $\widehat{k} \in \mathcal{K}$ tale che $\widehat{m} = Dec_{\widehat{k}}(c)$ (ricordando che abbiamo assunto Dec come deterministico). Sotto l'ipotesi $|\mathcal{K}| < |\mathcal{M}|$, si ha che esiste almeno un $m' \in \mathcal{M}$ tale che $m' \notin \mathcal{M}(c)$, allora:

$$Pr[M = m' \mid C = c] = 0 \neq Pr[M = m']$$

e questo ci dice che lo schema non ha sicurezza di Shannon. □

Notiamo che dalla dimostrazione del teorema 1.4.1 si deduce un possibile attacco contro ogni schema crittografico a chiave privata con $|\mathcal{K}| < |\mathcal{M}|$. Mostriamolo attraverso il seguente esempio:

Esempio 1.4. Consideriamo due parti oneste, Alice e Bob, le quali vogliono scambiarsi messaggi tramite un sistema crittografico che ha $|\mathcal{K}| < |\mathcal{M}|$. Consideriamo poi la presenza di un avversario, Eve, che vuole intercettare e decifrare i messaggi scambiati da Alice e Bob. Sia $k \in \mathcal{K}$ una chiave data dall'algoritmo Gen, esistono quindi $m_1, m_2 \in \mathcal{M}$ e una costante $\epsilon > 0$ tale che, dato un testo cifrato $c \in \mathcal{C}$ con $Enc_k(m_1) = c$ e $\mathcal{M}(c)$ definito come nella dimostrazione del teorema 1.4.1, valgono:

$$Pr[m_1 \in \mathcal{M}(c)] = 1$$

e

$$Pr[m_2 \in \mathcal{M}(c)] \leq 1 - \epsilon.$$

La prima equazione segue direttamente dalla definizione di schema crittografico a chiave privata, la seconda, invece, segue dal fatto che (per la dimostrazione del teorema 1.4.1) esiste qualche chiave per la quale $Enc_k(m_1) = c$, ma $m_2 \notin \mathcal{M}(c)$. Consideriamo ora uno scenario in cui Alice estrae, in modo uniforme, un messaggio m da $\{m_1, m_2\}$ e invia il relativo testo cifrato a Bob. Affermiamo che Eve, dopo aver visto il testo cifrato c corrispondente al testo in chiaro m, potrà indovinare se $m = m_1$ o $m = m_2$ con probabilità maggiore di $1/2$. Infatti Eve, dopo aver intercettato c, deve controllare semplicemente se $m_2 \in \mathcal{M}(c)$. Se $m_2 \notin \mathcal{M}(c)$ Eve saprà che $m = m_1$, altrimenti farà una supposizione casuale. Quindi se Alice inviasse il messaggio $m = m_2$, allora $m_2 \in \mathcal{M}(c)$ e Eve saprà indovinare il testo in chiaro con probabilità $1/2$. Se, d'altra parte, Alice inviasse il messaggio $m = m_1$, allora con probabilità ϵ

$m_2 \notin \mathcal{M}(c)$ e Eve saprà capire quale è il messaggio con probabilità 1. Con probabilità $1-\epsilon$, invece, $m_2 \in \mathcal{M}(c)$ e Eve dovrà fare una supposizione casuale, quindi avrà probabilità 1/2 di indovinare il messaggio corretto.
Concludiamo che la probabilità di successo di Eve è:

$$\frac{1}{2}Pr[m = m_2] + Pr[m = m_1](\epsilon \cdot 1 + (1-\epsilon)\cdot\frac{1}{2}) = \frac{1}{2} + \frac{\epsilon}{4} > \frac{1}{2}. \qquad (1.3)$$

Così abbiamo descritto un attacco conciso per Eve che le permette di capire quale messaggio Alice invia con probabilità maggiore di 1/2.

Questo attacco con un ϵ molto piccolo (per esempio 2^{-100}) potrebbe essere inefficiente in quanto la probabilità di successo di Eve diminuirebbe diventando circa 1/2. Nel prossimo teorema facciamo vedere che se la chiave è solamente un bit più corta rispetto al messaggio allora $\epsilon = 1/2$.

Teorema 1.4.2. *Sia $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ uno schema crittografico con $\mathcal{M} = \{0,1\}^n$ e $\mathcal{K} = \{0,1\}^{n-1}$. Allora esistono messaggi $m_1, m_2 \in \mathcal{M}$ tali che, data una chiave $k \in \mathcal{K}$ e $c \in \mathcal{C}$ con $Enc_k(m_1) = c$, vale:*

$$Pr[m_2 \in \mathcal{M}(c)] \leq \frac{1}{2},$$

dove $\mathcal{M}(c)$ è definito come nella dimostrazione del teorema 1.4.1.

Dimostrazione. Dato $c \leftarrow Enc_k(m)$ per qualche chiave $k \in \mathcal{K}$ e $m \in \mathcal{M}$, consideriamo l'insieme $\mathcal{M}(c)$. Dal fatto che abbiamo assunto Dec deterministico segue che $|\mathcal{M}(c)| \leq |\mathcal{K}| = 2^{n-1}$. Così, per ogni $m_1 \in \mathcal{M}$ e $k \in \mathcal{K}$, dato $c \in \mathcal{C}$ con $c = Enc_k(m_1)$, vale:

$$Pr[m' \in \mathcal{M}(c)] \leq \frac{2^{n-1}}{2^n} = \frac{1}{2}, \qquad m' \in \mathcal{M}.$$

Dal momento che la probabilità sopra è limitata da 1/2 per ogni chiave $k \in \mathcal{K}$, questo deve valere anche per una chiave casuale generata dall'algoritmo Gen. Inoltre, poichè questo limite vale per un messaggio casuale m', deve esistere qualche messaggio particolare m_2 che minimizzi la probabilità. In altre parole, considerata una chiave casuale $k \in \mathcal{K}$ data dall'algoritmo Gen, per ogni messaggio $m_1 \in \mathcal{M}$, con $c \in \mathcal{C}$ dato da $c = Enc_k(m_1)$, esiste qualche messaggio $m_2 \in \mathcal{M}$ tale che:

$$Pr[m_2 \in \mathcal{M}(c)] \leq \frac{1}{2}$$

e questo dimostra il teorema. □

Osservazione 6. Facendo riferimento all'attacco contro un sistema crittografico descritto nell'esempio 1.4, concludiamo, per il teorema 1.4.2, che la probabilità di successo di Eve (utilizzando la formula 1.3 con $\epsilon = 1/2$) è:

$$\frac{1}{2} + \frac{\frac{1}{2}}{4} = \frac{1}{2} + \frac{1}{8} = \frac{5}{8} > \frac{1}{2}.$$

Nel prossimo teorema daremo una caratterizzazione per schemi crittografici sicuri secondo Shannon. Come vedremo, questa caratterizzazione dice che, assumendo $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$, l'algoritmo di generazione delle chiavi, Gen, deve scegliere la chiave segreta in modo uniforme nell'insieme di tutte le chiavi possibili, e che per ogni testo in chiaro e testo cifrato deve esistere un'unica chiave che cifri il testo in chiaro nel testo cifrato. Questo teorema, dovuto a Claude Shannon, è uno strumento molto potente per dimostrare o confutare la sicurezza secondo Shannon di un dato sistema crittografico.

Come in precedenza, assumiamo che le distribuzioni di probabilità su $\mathcal{M}$ e $\mathcal{C}$ siano tali che ogni $m \in \mathcal{M}$ e ogni $c \in \mathcal{C}$ siano scelti con probabilità diversa da zero. Enunciamo ora il teorema:

Teorema 1.4.3 (Teorema di Shannon)**.** *Sia $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ uno schema crittografico con $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$. Lo schema è sicuro secondo Shannon se e solo se:*

i) ogni chiave $k \in \mathcal{K}$ viene scelta con uguale probabilità, $1/|\mathcal{K}|$, dall'algoritmo Gen;

ii) per ogni $m \in \mathcal{M}$ e ogni $c \in \mathcal{C}$, esiste un'unica chiave $k \in \mathcal{K}$ tale che $Enc_k(m) = c$.

Dimostrazione. Per semplicità, assumiamo che anche Enc sia un algoritmo deterministico. Supponiamo che $(\mathcal{M}, \mathcal{K}, Gen, Enc, Dec)$ sia uno schema che abbia sicurezza di Shannon e proviamo che valgono la i) e la ii). Come nella dimostrazione del teorema 1.4.1, non è difficile vedere che per ogni $m \in \mathcal{M}$ e per ogni $c \in \mathcal{C}$ esiste almeno una chiave $k \in \mathcal{K}$ tale che $Enc_k(m) = c$ (in caso contrario, $Pr[M = m \mid C = c] = 0 \neq Pr[M = m]$). Per un m fissato consideriamo l'insieme $\{Enc_k(m)\}_{k \in \mathcal{K}}$. Poichè per ogni $c \in \mathcal{C}$ esiste almeno un $k \in \mathcal{K}$ tale che $Enc_k(m) = c$, allora necessariamente $\left|\{Enc_k(m)\}_{k \in \mathcal{K}}\right| \geq |\mathcal{C}|$. Inoltre abbiamo che $\left|\{Enc_k(m)\}_{k \in \mathcal{K}}\right| \leq |\mathcal{C}|$ e quindi vale l'uguaglianza:

$$\left|\{Enc_k(m)\}_{k \in \mathcal{K}}\right| = |\mathcal{C}|\,.$$

Per ipotesi abbiamo che $|\mathcal{C}| = |\mathcal{K}|$, dunque $\left|\{Enc_k(m)\}_{k \in \mathcal{K}}\right| = |\mathcal{K}|$. Questo implica che non ci sono due chiavi distinte $k_1, k_2 \in \mathcal{K}$ tali che $Enc_{k_1}(m) =$

$Enc_{k_2}(m)$. Dall'arbitrarietà di m abbiamo mostrato che, per ogni $m \in \mathcal{M}$ e ogni $c \in \mathcal{C}$, esiste al più una chiave $k \in \mathcal{K}$ tale che $Enc_k(m) = c$. Combinando quanto sopra, cioè l'esistenza di almeno una chiave e al più una chiave, otteniamo che vale la *ii*). Rimane da dimostrare che per ogni $k \in \mathcal{K}$, $Pr[K = k] = 1/|\mathcal{K}|$. Sia $n = |\mathcal{K}|$ e $\mathcal{M} = \{m_1, \ldots, m_n\}$ (ricordando che $|\mathcal{M}| = |\mathcal{K}| = n$) e fissiamo anche un testo cifrato c. Allora possiamo chiamare le chiavi $k_1, \ldots, k_n$ in modo che per $j \in \{1, \ldots, n\}$ si ha $Enc_{k_j}(m_j) = c$. Tale associazione può essere eseguita perchè, come visto in precedenza, per ogni c e ogni m_j esiste un'unica chiave k_j tale che $Enc_{k_j}(m_j) = c$. Inoltre, queste chiavi sono distinte per distinti m_j, m_i. Dalla sicurezza secondo Shannon sappiamo che per ogni $j \in \{1, \ldots, n\}$, vale:

$$\begin{aligned} Pr[M = m_i] &= Pr[M = m_j \mid C = c] \\ &= \frac{Pr[C = c \mid M = m_j]Pr[M = m_j]}{Pr[C = c]} \\ &= \frac{Pr[K = k_j]Pr[M = m_j]}{Pr[C = c]}, \end{aligned}$$

dove la seconda uguaglianza deriva dal teorema 1.2 e la terza invece è conseguenza del fatto che, per l'associazione fatta in precedenza, k_j è l'unica chiave che mappa m_j con c. Per quanto appena visto abbiamo che per ogni j vale:

$$Pr[K = k_j] = Pr[C = c].$$

Perciò, per ogni j e i, $Pr[K = k_j] = Pr[C = c] = Pr[K = k_i]$ e quindi ogni chiave è scelta con la stessa probabilità. Possiamo allora concludere che le chiavi sono scelte in accordo alla distribuzione uniforme, cioè per ogni $k_j \in \mathcal{K}$, si ha che $Pr[K = k_j] = 1/|\mathcal{K}|$. Abbiamo così concluso la prima parte della dimostrazione.
Supponiamo ora che ogni chiave è ottenuta con probabilità $1/|\mathcal{K}|$ e che per ogni $m \in \mathcal{M}$ e $c \in \mathcal{C}$ esiste un unico $k \in \mathcal{K}$ tale che $Enc_k(m) = c$. Ciò implica che per ogni m e c si ha che:

$$Pr[C = c \mid M = m] = \frac{1}{|\mathcal{K}|}$$

indipendentemente dalla distribuzione di probabilità su $\mathcal{M}$. Così, per ogni distribuzione su $\mathcal{M}$, ogni $m, m' \in \mathcal{M}$ e ogni $c \in \mathcal{C}$ abbiamo:

$$Pr[C = c \mid M = m] = \frac{1}{|\mathcal{K}|} = Pr[C = c \mid M = m'],$$

cioè lo schema crittografico è perfettamente sicuro. Per quanto visto nel teorema 1.3.2 lo schema crittografico risulta essere sicuro anche secondo Shannon. □

In conclusione di questo capitolo mostriamo un esempio dove applichiamo la teoria studiata fin ora. In particolare, descriviamo un semplice sistema crittografico del quale analizziamo la sicurezza secondo le definizioni date in precedenza.

Esempio 1.5. Consideriamo il sistema crittografico definito da:

- $\mathcal{M} = \{a, b, c\}$ con $Pr[M = a] = 1/6$, $Pr[M = b] = 1/3$, $Pr[M = c] = 1/2$;
- $\mathcal{K} = \{k_1, k_2, k_3\}$ con $Pr[K = k_1] = 1/2$, $Pr[K = k_2] = Pr[K = k_3] = 1/4$;
- $\mathcal{C} = \{1, 2, 3\}$;
- L'algoritmo di cifratura Enc funziona nel modo seguente:
  $Enc_{k_1}(a) = 2$, $Enc_{k_1}(b) = 3$, $Enc_{k_1}(c) = 1$;
  $Enc_{k_2}(a) = 1$, $Enc_{k_2}(b) = 2$, $Enc_{k_2}(c) = 3$;
  $Enc_{k_3}(a) = 3$, $Enc_{k_3}(b) = 1$, $Enc_{k_3}(c) = 2$.

Calcolando la distribuzione di probabilità su $\mathcal{C}$ otteniamo i seguenti valori:

$$Pr[C = 1] = \frac{1}{4} + \frac{1}{24} + \frac{1}{12} = \frac{3}{8};$$

$$Pr[C = 2] = \frac{1}{12} + \frac{1}{12} + \frac{1}{8} = \frac{7}{24};$$

$$Pr[C = 3] = \frac{1}{6} + \frac{1}{8} + \frac{1}{24} = \frac{1}{3}.$$

Ora calcoliamo le distribuzioni di probabilità condizionata sul testo in chiaro, osservato un dato testo cifrato. Abbiamo:

$$Pr[M = a \mid C = 1] = \frac{1}{9}, \quad Pr[M = b \mid C = 1] = \frac{2}{9}, \quad Pr[M = c \mid C = 1] = \frac{2}{3};$$

$$Pr[M = a \mid C = 2] = \frac{2}{7}, \quad Pr[M = b \mid C = 2] = \frac{2}{7}, \quad Pr[M = c \mid C = 2] = \frac{3}{7};$$

$$Pr[M = a \mid C = 3] = \frac{1}{8}, \quad Pr[M = b \mid C = 3] = \frac{1}{2}, \quad Pr[M = c \mid C = 3] = \frac{3}{8}.$$

Possiamo concludere che questo schema crittografico non è sicuro secondo Shannon, infatti abbiamo che:

$$Pr[M = a \mid C = 1] = \frac{1}{9} \neq \frac{1}{6} = Pr[M = a],$$

cioè non è verificata la condizione $Pr[M = m \mid C = c] = Pr[M = m]$ che per avere un crittosistema sicuro secondo Shannon deve valere per ogni testo in chiaro e per ogni testo cifrato.

Osservazione 7. Nell'esempio precedente abbiamo dimostrato che il crittosistema non è sicuro secondo Shannon utilizzando direttamente la definizione 1.4; questo è stato possibile solo dopo aver calcolato le distribuzioni di probabilità e di probabilità condizionata. Nel crittosistema abbiamo che lo spazio dei messaggi, quello dei testi in chiaro e quello delle chiavi siano "piccoli", e questo ci ha permesso di calcolare le varie distribuzioni in modo rapido. Nel caso in cui lo spazio dei messaggi, quello dei testi cifrati e quello delle chiavi siano notevolmente più grandi, il calcolo delle distribuzioni potrebbe risultare lungo e laborioso e renderebbe difficile la dimostrazione della sicurezza di Shannon del crittosistema, utilizzando la definizione 1.4. Abbiamo dimostrato precedentemente alcuni teoremi grazie ai quali potevamo capire che il sistema crittografico non era sicuro secondo Shannon senza dover calcolare le distribuzioni di probabilità e di probabilità condizionata. Infatti, notando che nel crittosistema dell'esempio si ha $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$, allora possiamo applicare il teorema di Shannon il quale ci dice che un sistema crittografico con $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$ è sicuro secondo Shannon se e solo se ogni chiave $k \in \mathcal{K}$ è scelta con uguale probabilità. Questa condizione non è verificata dal crittosistema preso in esame. Infatti, nella definizione del cifrario è stato assunto che la chiave k_1 è scelta con probabilità uguale a $1/2$, mentre le chiavi k_2 e k_3 sono scelte con probabilità pari a $1/4$.

Capitolo 2

Sicurezza computazionale

Nel capitolo precedente abbiamo definito la sicurezza di Shannon per sistemi crittografici a chiave privata. Come è stato già osservato, i sistemi crittografici sicuri secondo Shannon presentano alcuni problemi, come per esempio la lunghezza delle chiavi, che non permettono di poter usare questi cifrari nella pratica. Solitamente, infatti, si utilizzano cifrari che sono sicuri supponendo che un ipotetico avversario abbia una capacità computazionale limitata. In questo modo all'avversario serviranno tempi molto lunghi per forzare il sistema crittografico con la potenza di calcolo a sua disposizione. Questo concetto di sicurezza è riassunto molto bene da uno dei sei principi di Kerckhoffs: *Un cifrario deve essere praticamente, se non matematicamente, indecifrabile.*

In seguito analizziamo in dettaglio questo tipo di sicurezza che chiamiamo *sicurezza computazionale.*

2.1 Efficienza di calcolo e avversario efficiente

Con *efficienza di calcolo (o computazionale)* intendiamo la capacità di un *algoritmo* di utilizzare meno risorse informatiche possibili. Principalmente vengono considerati due fattori:

- il tempo di utilizzo della CPU;[1]
- lo spazio occupato dall'algoritmo e dai dati in memoria.

[1]CPU indica l'unità di elaborazione centrale, cioè una tipologia di processore digitale general purpose, la quale si contraddistingue per sovrintendere tutte le funzionalità del computer digitale.

L'efficienza di calcolo, quindi, si basa fondamentalmente sull'efficienza degli algoritmi che si hanno a disposizione. Diamo ora alcune definizioni che ci permettono di comprendere il significato di algoritmo efficiente.

Definizione 2.1 (Algoritmo)**.** Un algoritmo è una macchina di Turing[2] deterministica che prende in input stringhe sull'alfabeto $\Sigma = \{0, 1\}$, e restituisce ancora stringhe sull'alfabeto Σ.

Definizione 2.2 (Tempo d'esecuzione)**.** Un algoritmo $\mathcal{A}$ si dice eseguito in tempo $T(n)$ se per ogni $x \in \{0,1\}^*$, $\mathcal{A}(x)$ si arresta in meno di $T(|x|)$ passi, dove con $|x|$ indichiamo la lunghezza della stringa x. $\mathcal{A}$ viene eseguito in tempo polinomiale se esiste un polinomio $p(\cdot)$ tale che per ogni $x \in \{0,1\}^*$, $\mathcal{A}(x)$ si arresta in un numero di passi minore o uguale a $p(|x|)$.

In altre parole, un algoritmo è eseguito in tempo polinomiale se esistono due costanti a, r tale che $\mathcal{A}(x)$ termina in $a \cdot |x|^r$ passi.

Definizione 2.3 (Algoritmo efficiente)**.** Un algoritmo $\mathcal{A}$ è detto efficiente se è eseguito in tempo polinomiale.

Osservazione 8. Usare il tempo polinomiale come limite massimo per definire l'efficienza di un algoritmo potrebbe essere soggetto a obiezioni. In effetti, se il grado del polinomio in questione è abbastanza grande, il calcolo può non essere efficiente in pratica. Tuttavia, l'esperienza suggerisce che gli algoritmi in tempo polinomiale risultano essere efficienti, cioè tempo polinomiale significa quasi sempre "tempo cubico o migliore".

Definizione 2.4 (Calcolo deterministico)**.** Si dice che un algoritmo $\mathcal{A}$ calcola una funzione $f : \{0,1\}^* \longrightarrow \{0,1\}^*$, se per ogni $x \in \{0,1\}^*$, $\mathcal{A}$ con input x, restituisce come output $f(x)$.

Una naturale estensione della nozione di calcolo deterministico è consentire a un algoritmo di avere accesso a una fonte di "lanci di monete" casuali. Quest'ulteriore condizione serve per consentire agli algoritmi una maggiore efficienza nel calcolo di alcuni compiti e sarà necessaria per la sicurezza dei sistemi crittografici che presenteremo in seguito. Per esempio, come abbiamo visto nel primo capitolo, uno dei principi di Kerckhoffs afferma che tutti gli algoritmi di un sistema di crittografia dovrebbero essere pubblici. Così, se l'algoritmo di generazione delle chiavi private *Gen* non ha utilizzato il lancio delle monete casuali per il calcolo della chiave, allora l'avversario Eve sarebbe

[2]Una macchina di Turing è una macchina ideale che manipola i dati contenuti su un nastro di lunghezza potenzialmente infinita, secondo un insieme prefissato di regole ben definite.

in grado di calcolare la stessa chiave che le parti oneste, Alice e Bob, hanno calcolato. Estendiamo quindi la definizione di calcolo deterministico come segue:

Definizione 2.5 (Algoritmo probabilistico)**.** Un algoritmo probabilistico, anche chiamato macchina di Turing probabilistica in tempo polinomiale, abbreviato con **PPT** (dall'inglese probabilistic polynomial-time), è una macchina di Turing dotata di un nastro casuale supplementare. Ogni bit del nastro casuale viene scelto uniformemente e in modo indipendente.

Equivalentemente, un algoritmo probabilistico è una macchina di Turing che ha accesso a un *oracolo casuale* (lancio di moneta) che genera, su richiesta, un bit casuale.

Per definire la nozione di efficienza di calcolo dobbiamo chiarire il concetto di tempo di esecuzione di un algoritmo probabilistico. Definiremo il tempo di esecuzione di un algoritmo probabilistico come il limite superiore su tutti i nastri casuali, perchè le sequenze casuali di cui è dotato l'algoritmo probabilistico possono far variare il tempo della sua esecuzione.

Definizione 2.6 (Tempo di esecuzione di un algoritmo probabilistico)**.** Una macchina di Turing probabilistica $\mathcal{A}$ viene eseguita in tempo $T(n)$ se per ogni $x \in \{0,1\}^*$ e per ogni nastro casuale, $\mathcal{A}(x)$ si arresta in un numero di passi minore o uguale a $T(|x|)$. Diremo che $\mathcal{A}$ è eseguito in tempo polinomiale (oppure $\mathcal{A}$ è un algoritmo probabilistico efficiente) se esistono due costanti a, r tali che $\mathcal{A}(x)$ è eseguito in tempo $T(|x|) = a \cdot |x|^r$ per ogni $x \in \{0,1\}^*$.

Infine, dobbiamo anche estendere la nozione di computazione ad algoritmi probabilistici. In particolare, una volta che un algoritmo ha un nastro casuale, il suo output diventa una distribuzione su qualche insieme. Nel caso del calcolo deterministico l'output è un insieme costituito da un singolo elemento.

Definizione 2.7. Si dice che un algoritmo probabilistico $\mathcal{A}$ calcola una funzione $f : \{0,1\}^* \longrightarrow \{0,1\}^*$, se per ogni $x \in \{0,1\}^*$, $\mathcal{A}$ con input x, restituisce come output $f(x)$ con probabilità 1.

Notiamo che alcuni algoritmi probabilistici usati nella pratica (ad esempio i test di primalità[3]), con probabilità molto basse, possono commettere errori. Nel seguito del capitolo ignoriamo questi rari casi e supponiamo che gli algoritmi probabilistici funzionino sempre correttamente.

[3]Un test di primalità è un algoritmo che, applicato ad un numero intero, ha lo scopo di determinare se esso è primo. I test di primalità più efficienti sono probabilistici.

In definitiva utilizziamo come modello di efficienza di calcolo gli algoritmi probabilistici in tempo polinomiale. Facciamo riferimento a questa classe di algoritmi come macchine di Turing probabilistiche in tempo polinomiale (**PPT**) o, indifferentemente, come algoritmi probabilistici efficienti.

Diamo ora la definizione di *schema crittografico a chiave privata efficiente.*

Definizione 2.8 (Schema crittografico a chiave privata efficiente)**.** La terna (Gen, Enc, Dec) si dice uno schema crittografico a chiave privata efficiente se vale quanto segue:

1. Gen è un **PPT** tale che, per ogni $n \in \mathbb{N}$, $Gen(1^n)$ genera una chiave k;
2. Enc è un **PPT** che data una chiave k e un messaggio $m \in \{0,1\}^n$ restituisce un testo cifrato c, cioè si ha che $Enc_k(m) = c$;
3. Dec è un **PPT** che data una chiave k e un testo cifrato c produce un messaggio $m \in \{0,1\}^n$, cioè si ha che $Dec_k(c) = m$;
4. per ogni $n \in \mathbb{N}$, $m \in \{0,1\}^n$, data una chiave k generata da $Gen(1^n)$, si ha:
$$Pr[Dec_k(Enc_k(m)) = m] = 1.$$

Nella definizione appena data notiamo che l'algoritmo Gen ha come input 1^n. Questo input particolare indica una stringa di n copie di 1 e viene chiamato *parametro di sicurezza*; più questo parametro è grande e maggiore sarà la sicurezza dello schema. Il parametro di sicurezza stabilisce inoltre il tempo di esecuzione dell'algoritmo Gen, quindi la lunghezza della chiave k e, di conseguenza, anche i tempi di esecuzione di Enc e Dec.
A differenza della definizione di schema crittografico a chiave privata data nel primo capitolo, nella definizione precedente non si fa più riferimento allo spazio dei messaggi $\mathcal{M}$ e a quello delle chiavi $\mathcal{K}$ perchè si suppone che i testi in chiaro e le chiavi siano stringhe di bit. In particolare, con il parametro di sicurezza 1^n, la definizione di schema crittografico a chiave privata efficiente descrive uno schema che gestisce messaggi lunghi n bit. Nel seguito con sistema crittografico a chiave privata ci riferiremo a sistemi crittografici efficienti nel senso della definizione precedente.

La nozione di sicurezza computazionale prende in considerazione anche la potenza di calcolo di un ipotetico avversario; naturalmente un sistema di crittografia a chiave privata deve essere computazionalmente sicuro di fronte a degli *avversari efficienti.* Rappresentiamo gli avversari efficienti sempre tramite degli algoritmi, e come in precedenza l'avversario dispone di una fonte di "lanci di monete" casuali, inoltre supponiamo che il tempo di esecuzione dell'algoritmo sia polinomiale. Rappresentiamo sempre gli avversari efficienti tramite delle **PPT**.

2.2 Funzioni unidirezionali

Nella crittografia moderna si richiede che gli schemi crittografici efficienti abbiano le seguenti caratteristiche:

- deve essere possibile calcolare un testo cifrato c data una chiave k e un messaggio m;
- deve essere "difficile" calcolare m e k conoscendo solamente il testo cifrato c.

Si richiede quindi l'utilizzo di funzioni che sono facili da calcolare ma difficili da invertire; questo tipo di funzioni le chiameremo *funzioni unidirezionali*. Ci sono diversi modi per definire questa tipologia di funzioni, ma nel seguito daremo solo la definizione di *funzione unidirezionale forte*, la più indicata per i nostri scopi. Per questa definizione abbiamo bisogno del concetto di *funzione trascurabile*:

Definizione 2.9 (Funzione trascurabile)**.** Una funzione $\epsilon(\cdot)$ è detta trascurabile se, per ogni r esiste n_0 tale che per ogni $n > n_0$, vale:

$$\epsilon(n) \leq \frac{1}{n^r}.$$

Definizione 2.10 (Funzione unidirezionale forte)**.** Una funzione

$$f : \{0,1\}^* \longrightarrow \{0,1\}^*$$

è detta funzione unidirezionale forte se soddisfa le seguenti condizioni:

1. è facile da calcolare, cioè esiste una **PPT** che calcola $f(x)$ per tutti gli input $x \in \{0,1\}^*$;
2. è difficile da invertire, cioè ogni tentativo efficiente per invertire la funzione f con input casuale ha successo solo con probabilità trascurabile. Formalmente, per qualsiasi avversario efficiente $\mathcal{A}$, esiste una funzione trascurabile $\epsilon(\cdot)$ tale che per ogni input $x \in \{0,1\}^n$ di lunghezza $n \in \mathbb{N}$ si ha che:
$$Pr[f(\mathcal{A}(1^n, f(x)) = f(x)] \leq \epsilon(n).$$

Notiamo che l'algoritmo $\mathcal{A}$ riceve l'input aggiuntivo 1^n, questo fa si che il tempo d'esecuzione dell'algoritmo sia polinomiale in $|x| = n$.

2.3 Indistinguibilità e pseudo-casualità

Nel primo capitolo abbiamo analizzato il cifrario di Vernam, il quale ha una semplice operazione di cifratura, $Enc_k(m) = m \oplus k$, ma presenta il grande problema della lunghezza della chiave che deve essere lunga almeno quanto il messaggio. Per rendere il sistema più efficiente si può pensare di iniziare con una breve chiave casuale k, quindi provare ad utilizzare un generatore pseudo-casuale G per espanderla a una chiave più lunga, $k' = G(k)$, "dall'aspetto casuale" e infine utilizzare il cifrario di Vernam con la chiave k'. Osserviamo che non possono esistere generatori pseudo-casuali che con input k generano una stringa perfettamente casuale k', in quanto ciò sarebbe in contraddizione con il teorema 1.4.1 dove si supponeva che l'avversario avesse capacità computazionale illimitata. Spostando però la nostra attenzione solo su avversari efficienti, che hanno capacità computazionale limitata, possiamo riuscire a mettere a punto generatori pseudo-casuali che restituiscono stringhe, "dall'aspetto sufficientemente casuale", adatti alle applicazioni crittografiche. Il concetto di stringa "dall'aspetto sufficientemente casuale" non è facile da formalizzare, ma cerchiamo comunque di dare una definizione intuitiva di questa nozione. Una stringa può essere considerata una stringa "dall'aspetto casuale" se soddisfa condizioni del tipo:

- ha approssimativamente tanti 0 quanti 1;
- ha approssimativamente tanti 00 quanti 11.

Verificare se una stringa soddisfa le condizioni precedenti significa effettuare un *test statistico*[4] sulla stringa stessa. Diciamo quindi che una stringa ha "un aspetto sufficientemente casuale" se passa alcuni test statistici.

Indistinguibilità computazionale. Introduciamo la nozione di *indistinguibilità computazionale* che formalizza il concetto che due distribuzioni di probabilità possono apparire la stessa dal punto di vista di un avversario computazionalmente limitato. Questo concetto è uno dei cardini della crittografia moderna.

La definizione di indistinguibilità computazionale necessita dell'uso di sequenze, chiamate insiemi, di distribuzioni di probabilità.

Definizione 2.11 (Insieme di distribuzioni di probabilità)**.** Una sequenza $\{X_n\}_{n\in\mathbb{N}}$ è chiamata un insieme di distribuzioni di probabilità se per ogni $n \in \mathbb{N}$, X_n è una distribuzione di probabilità su $\{0,1\}^*$.

[4] Un test statistico è una regola di decisione. Effettuare un test statistico su una stringa di bit significa verificare se la stringa soddisfa delle determinate ipotesi (per esempio la presenza di tanti 0 quanti 1).

Definizione 2.12 (Indistinguibilità computazionale)**.** Siano $X = \{X_n\}_{n\in\mathbb{N}}$ e $Y = \{Y_n\}_{n\in\mathbb{N}}$ due insiemi di distribuzioni di probabilità con X_n e Y_n distribuzioni su $\{0,1\}^{l(n)}$ per un qualche polinomio $l(\cdot)$. Allora diremo che X e Y sono computazionalmente indistinguibili, in simboli $X \approx Y$, se per ogni **PPT** D, chiamata *distinguisher*, esiste una funzione trascurabile $\epsilon(\cdot)$ tale che:

$$|Pr[D(1^n, X_n) = 1] - Pr[D(1^n, Y_n) = 1]| \leq \epsilon(n),$$

dove la notazione $D(1^n, X_n)$ indica che viene scelto un x secondo la distribuzione X_n e poi viene eseguito $D(1^n, x)$.

Osserviamo che al *distinguisher* D è dato anche l'input unario 1^n, così che può essere eseguito in tempo polinomiale in n. Per semplificare la notazione diremo che D distingue la distribuzione X_n da Y_n con probabilità ϵ, se:

$$|Pr[D(1^n, X_n) = 1] - Pr[D(1^n, Y_n) = 1]| > \epsilon.$$

Inoltre diremo che D distingue gli insiemi di distribuzioni di probabilità $\{X_n\}_{n\in\mathbb{N}}$ e $\{Y_n\}_{n\in\mathbb{N}}$ con probabilità $\mu(\cdot)$ se, per ogni $n \in \mathbb{N}$, D distingue X_n da Y_n con probabilità $\mu(n)$.

Analizziamo ora due importanti proprietà dell'indistinguibilità computazionale. La prima dice che se due distribuzioni sono indistinguibili, allora rimangono indistinguibili anche dopo aver applicato loro una **PPT**.

Lemma 2.3.1. *Siano $\{X_n\}_{n\in\mathbb{N}}$ e $\{Y_n\}_{n\in\mathbb{N}}$ due insiemi di distribuzioni di probabilità. Se $\{X_n\}_{n\in\mathbb{N}} \approx \{Y_n\}_{n\in\mathbb{N}}$, allora per ogni **PPT** M, $\{M(X_n)\}_{n\in\mathbb{N}} \approx \{M(Y_n)\}_{n\in\mathbb{N}}$.*

Dimostrazione. Supponiamo che esista una **PPT** D e una funzione non trascurabile $\mu(\cdot)$ tale che D distingua $\{M(X_n)\}_{n\in\mathbb{N}}$ da $\{M(Y_n)\}_{n\in\mathbb{N}}$ con probabilità $\mu(n)$, cioè:

$$|Pr[D(1^n, M(X_n)) = 1] - Pr[D(1^n, M(Y_n)) = 1]| > \mu(n),$$

dove la notazione $D(1^n, M(X_n))$ indica che viene scelto un x secondo la distribuzione $M(X_n)$ e poi viene eseguito $D(1^n, x)$. Dall'ultima disuguaglianza sappiamo quindi che presi x e y, dati rispettivamente dalle distribuzioni X_n e Y_n, vale:

$$|Pr[D(1^n, D(M(x))) = 1] - Pr[D(1^n, D(M(y))) = 1]| > \mu(n).$$

In questo caso, la **PPT** $D'(\cdot) = D(M(\cdot))$ distingue anche $\{X_n\}_{n\in\mathbb{N}}$ e $\{Y_n\}_{n\in\mathbb{N}}$ con probabilità $\mu(n)$ contraddicendo l'ipotesi che $\{X_n\}_{n\in\mathbb{N}} \approx \{Y_n\}_{n\in\mathbb{N}}$. □

La seconda proprietà dice che l'indistinguibilità computazionale è transitiva, cioè se $\{X_n\}_{n\in\mathbb{N}} \approx \{Y_n\}_{n\in\mathbb{N}}$ e $\{Y_n\}_{n\in\mathbb{N}} \approx \{Z_n\}_{n\in\mathbb{N}}$, allora $\{X_n\}_{n\in\mathbb{N}} \approx \{Z_n\}_{n\in\mathbb{N}}$. Dimostriamo quindi una generalizzazione di questa affermazione che considera $m = l(n)$ distribuzioni, con $l(\cdot)$ polinomio.

Lemma 2.3.2 (Lemma ibrido)**.** *Sia $X_1, X_2, \ldots, X_m$ una sequenza di distribuzioni di probabilità. Supponiamo che la macchina D distingua X_1 da X_m con probabilità ϵ. Allora esiste qualche $i \in \{1, \ldots, m-1\}$ tale che D distingue X_i da X_{i+1} con probabilità ϵ/m.*

Dimostrazione. Supponiamo che D distingua X_1 da X_m con probabilità ϵ, cioè:

$$|Pr[D(X_1) = 1] - Pr[D(X_m) = 1]| > \epsilon.$$

Ricordiamo che con $D(X_1)$ intendiamo che viene preso un x secondo la distribuzione X_1 e poi viene eseguito $D(x)$. Usando sempre questa notazione sia $g_i = Pr[D(X_i) = 1]$, allora $|g_1 - g_m| > \epsilon$ e questo implica che:

$$\begin{aligned} &|g_1 - g_2| + |g_2 - g_3| + \cdots + |g_{m-1} - g_m| \\ &\geq |g_1 - g_2 + g_2 - g_3 + \cdots + g_{m-1} - g_m| \\ &= |g_1 - g_m| > \epsilon. \end{aligned}$$

Pertanto, deve esistere i tale che $|g_i - g_{i+1}| > \epsilon/m$, cioè tale che:

$$|Pr[D(X_i) = 1] - Pr[D(X_{i+1}) = 1]| > \frac{\epsilon}{m},$$

e questo implica che D distingue X_i da X_{i+1} con probabilità ϵ/m dimostrando così la tesi. □

Useremo più avanti queste due importanti proprietà dell'indistinguibilità computazionale per dimostrare la sicurezza di un particolare schema di crittografia.

Generatori pseudo-casuali. La pseudo-casualità è solo un caso particolare dell'indistinguibilità computazionale. Con $U_{l(n)}$ indichiamo la distribuzione uniforme su $\{0,1\}^{l(n)}$; abbiamo quindi la seguente definizione:

Definizione 2.13. Un insieme di distribuzione di probabilità $X = \{X_n\}_{n\in\mathbb{N}}$ è pseudo-casuale se per qualche polinomio $l(\cdot)$, X è computazionalmente indistinguibile dall'insieme di distribuzioni di probabilità $U = \left\{U_{l(n)}\right\}_{n\in\mathbb{N}}$.

Possiamo ora definire i *generatori pseudo-casuali*, che chiameremo **PRG** (dall'inglese pseduo-random generator).

Definizione 2.14. Sia $l(\cdot)$ un polinomio e sia G un algoritmo deterministico eseguito in tempo polinomiale e supponiamo che per ogni s si abbia $|G(s)| = l(|s|)$. Diremo che G è un **PRG** se valgono le seguenti condizioni:

1. (espansione) per ogni n si ha che $l(n) > n$;
2. (pseudo-casualità) l'insieme $\{G(U_n)\}_{n\in\mathbb{N}}$ è pseudo-casuale.

Con la notazione $G(U_n)$ intendiamo che è stato scelto un u secondo la distribuzione U_n e poi eseguito $G(u)$.

2.4 Definizione computazionale di sicurezza

In quest'ultimo paragrafo useremo il concetto di indistinguibilità computazionale per dare la definizione di schema crittografico sicuro computazionalmente. Inoltre vedremo come i **PRG** sono indispensabili per la costruzione di schemi di crittografia che consentono l'uso di chiavi brevi per cifrare messaggi lunghi.

A differenza della definizione di sicurezza perfetta (equivalente alla sicurezza di Shannon), dove si richiede che la distribuzione dei testi cifrati per qualsiasi coppia di messaggi sia identica, nella definizione computazionale di sicurezza, invece, si richiede semplicemente che i testi cifrati di due messaggi qualsiasi siano indistinguibili.

Definizione 2.15. Uno schema crittografico (efficiente), (Gen, Enc, Dec), si dice *single-message secure* se per ogni **PPT** D, esiste una funzione trascurabile $\epsilon(\cdot)$ tale che per ogni $n \in \mathbb{N}$ e per ogni coppia di messaggi $m_1, m_2 \in \{0,1\}^n$, D distingue le distribuzioni:

- $\{Enc_k(m_1),\ con\ k\ data\ da\ Gen(1^n)\}$;
- $\{Enc_k(m_2),\ con\ k\ data\ da\ Gen(1^n)\}$,

con probabilità al più $\epsilon(n)$.

La definizione appena data si basa sull'indistinguibilità delle distribuzioni di testi cifrati, creati cifrando due diversi messaggi.

Abbiamo visto nel primo capitolo che i sistemi crittografici sicuri secondo Shannon richiedono chiavi lunghe almeno quanto i messaggi. Ora mostriamo come una chiave breve può essere utilizzata per costruire un sistema di crittografia sicuro secondo la definizione precedente. L'idea di base è quella di usare un cifrario di Vernam in cui una chiave breve viene espansa tramite un **PRG**, cosi da poter cifrare messaggi lunghi. Mostriamo in dettaglio tale schema crittografico nel seguente esempio:

Esempio 2.1. Consideriamo un **PRG** G che raddoppia la lunghezza dell'input. Definiamo lo schema crittografico (Gen, Enc, Dec), per la cifratura di messaggi lunghi n bit, nel seguente modo:

1. l'algoritmo Gen genera una chiave secondo la distribuzione $U_{\frac{n}{2}}$;
2. la cifratura avviene tramite l'algoritmo Enc come segue :
$$Enc_k(m) = m \oplus G(k);$$
3. la decifrazione tramite l'algoritmo Dec è data da :
$$Dec_k(c) = c \oplus G(k).$$

Come in precedenza $U_{\frac{n}{2}}$ indica la distribuzione uniforme su $\{0,1\}^{\frac{n}{2}}$.

Teorema 2.4.1. *Lo schema crittografico descritto nell'esempio 2.1 è single-message secure.*

Dimostrazione. Supponiamo per assurdo che esiste un *distinguisher* D e un polinomio $l(\cdot)$ tale che per infiniti n, esistono messaggi m_n^1, m_n^2 tali che D distingue le due distribuzioni di probabilità:

- $\{Enc_k(m_n^1)\}$, k generata da $Gen(1^n)$;
- $\{Enc_k(m_n^2)\}$, k generata da $Gen(1^n)$,

con probabilità $1/l(n)$. Consideriamo le seguenti distribuzioni ibride:

- cifrature di m_n^1: $H_n^1 = \{m_n^1 \oplus G(s)\}$, k generata da $Gen(1^n)$;
- Vernam con m_n^1: $H_n^2 = \{m_n^1 \oplus r\}$, r dato da U_n;
- Vernam con m_n^2: $H_n^3 = \{m_n^2 \oplus r\}$, r dato da U_n;
- cifrature di m_n^2: $H_n^4 = \{m_n^2 \oplus G(s)\}$, k generata da $Gen(1^n)$.

Per costruzione D distingue H_n^1 da H_n^4 con probabilità $1/l(n)$ per infiniti n. Segue dal lemma 2.3.2 che D distingue anche due distribuzioni ibride consecutive con probabilità $1/4l(n)$ per infiniti n. Mostriamo che questo dà luogo a una contraddizione. Infatti, consideriamo le **PPT** $M^i(x) = m^i_{|x|} \oplus x$ per $i = 1, 2$ e le distribuzioni $X_n = \left\{G(s),\ con\ s\ data\ da\ U_{\frac{n}{2}}\right\}$. Abbiamo quindi che $H_n^1 = M^1(X_n)$, $H_n^4 = M^2(X_n)$ e $H_n^2 = M^1(U_n)$, $H_n^3 = M^2(U_n)$. Essendo G un **PRG** abbiamo anche che $\{X_n\}_{n\in\mathbb{N}} \approx \{U_n\}_{n\in\mathbb{N}}$, allora dal lemma 2.3.1 segue che $\{H_n^1\}_{n\in\mathbb{N}} \approx \{H_n^2\}_{n\in\mathbb{N}}$ e $\{H_n^3\}_{n\in\mathbb{N}} \approx \{H_n^4\}_{n\in\mathbb{N}}$. Inoltre, essendo il cifrario di Vernam perfettamente sicuro, H_n^2 e H_n^3 sono identicamente distribuiti. Così, tutte le distribuzioni ibride consecutive sono indistinguibili e si ha quindi una contraddizione. □

Conclusione

Durante questo elaborato abbiamo analizzato due tipologie di sicurezza dei sistemi crittografici a chiave privata: la sicurezza di Shannon e la sicurezza computazionale. Abbiamo inoltre definito la sicurezza perfetta di un cifrario e abbiamo dimostrato che è equivalente alla sicurezza di Shannon. Nel primo capitolo, dopo aver esaminato in dettaglio la sicurezza di Shannon, abbiamo notato che si possono costruire sistemi crittografici sicuri secondo questa definizione, come per esempio il cifrario di Vernam, ma che risultano non realizzabili nella pratica. Per questo motivo, nel secondo capitolo, abbiamo introdotto la nozione di sicurezza computazionale. Questo tipo di sicurezza permette la costruzione di schemi crittografici che sono sicuri di fronte ad avversari che hanno una capacità computazionale limitata. Abbiamo rivolto successivamente la nostra attenzione a sistemi crittografici *single-message secure*, cioè sistemi crittografici sicuri per la cifratura di un singolo messaggio. Per poter dare la definizione formale di cifrario *single-message secure* è stato indispensabile definire le funzioni unidirezionali, l'indistinguibilità computazionale e i generatori pseudo-casuali. Infine, abbiamo costruito un sistema crittografico efficiente dimostrando che questo è *single-message secure*.

Bibliografia

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

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

[3] Rafael Pass, Abhi Shelat, *A Course in Cryptography*, disponibile al sito www.cs.cornell.edu/courses/cs4830/2008fa/lecnotes.pdf.

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

[5] Douglas R. Stinson, *Cryptography, Theory and Practice - Third Edition*, Chapman & Hall/CRC, 2006.