# Funzioni Pseudo-Random



## Introduzione

Le funzioni pseudo-random (PRF) e le permutazioni pseudo-random (PRP) sono tra le primitive più importanti della crittografia, specialmente per quella simmetrica. Essi sono particolarmente utili per analizzare la sicurezza dei cifrari a blocchi o meglio dei protocolli su questi basati.

## Famiglie di funzioni

Una **famiglia di funzioni** è una *mappa* del tipo:

$$F : K \times D \to R$$

dove:

- $K$ è l'insieme delle chiavi di $F$;
- $D$ è il dominio di $F$;
- $R$ è il range di $F$, ovvero il codominio.

Tutti gli insiemi sono finiti e non vuoti.

La funzione $F$ prende in input due valori: $k$ che è la chiave e $X$ che è il messaggio. Essa ritorna un valore $Y$ che definiamo con $F(k, X)$.
Per ogni chiave $k \in K$ è possibile definire la mappa $F_k : D \to R$ tale che $F_k(X) = F(k, X)$.
Chiamiamo la funzione $F_k$ un'*istanza* della famiglia di funzioni $F$. Quindi, $F$ specifica una collezione di mappe, una per ogni chiave. Proprio per questo motivo $F$ viene chiamata *famiglia di funzioni*.

## Parametri e annotazioni

- $K = \{0,1\}^k$, dove $k$ è un intero che stabilisce la lunghezza della chiave;
- $D = \{0,1\}^l$, dove $l$ è un intero che stabilisce la lunghezza dell'input;
- $R = \{0,1\}^L$, dove $L$ è un intero che stabilisce la lunghezza dell'output.

C'è una certa distribuzione di probabilità sull'insieme delle chiavi $K$. A meno che non sia indicato, questa distribuzione sarà quella uniforme. Si definisce con $a \leftarrow_R A$ l'operazione di scelta casuale di un elemento in A. In poche parole, sia $f$ la funzione $F_k$, dove $k$ è una chiave scelta a caso. Siamo interessati al comportamento input-output di questa istanza scelta casualmente dalla famiglia di funzioni.

**Definizione di permutazione**

Una permutazione è una biiezione (cioè un uno a uno sulla mappa) il cui dominio e codominio sono lo stesso insieme. Cioè, una mappa $\pi : D \to D$ è una permutazione se $\forall y \in D, \exists_1 x \in D \mid \pi(x) = y$. Quindi $F$ è una famiglia di permutazioni se $D = R$ e ogni $F_k$ è una permutazione su questo insieme comune.

**Esempi:**

- DES è una famiglia di permutazione ($k = 56, l = L = 64$);
- AES è una famiglia di permutazione ($k = l = L = 128$).

# Funzioni random e permutazioni

Siano $D, R \subseteq \{0,1\}^*$ insiemi finiti e non vuoti e sia $l, L \geq 1$ interi. Ci sono due particolari famiglie di funzioni che potranno essere considerate:

- $Func(D, R)$, la famiglia di tutte le funzioni da $D$ a $R$.
- $Perm(D)$, la famiglia di tutte le permutazioni su $D$.

Una istanza casuale di $Func(D, R)$ è dunque una funzione casuale da $D$ a $R$ e una istanza casuale di $Perm(D)$ è dunque una permutazione casuale su $D$.

## Funzioni casuali

La famiglia $Func(D, R)$ ha dominio $D$ e codominio $R$. L'insieme delle istanze di $Func(D, R)$ è l'insieme di tutte le funzioni che mappano $D$ a $R$. La chiave, che descrive ogni istanza, è la mera descrizione della funzione.

Per esempio, ordiniamo in maniera lessicografica il dominio $D$ come $X_1, X_2, \ldots$, e quindi lascia che la chiave per una funzione $f$ sia l'elenco dei valori $(f(X_1), f(X_2), \ldots)$. Lo spazio delle chiavi di $Func(D, R)$ è semplicemente l'insieme di tutte le chiavi.

Illustriamo più in dettaglio per il caso di $Func(l, L)$. La chiave per una funzione in questa famiglia è semplicemente una lista di tutti i valori di output della funzione quando i suoi intervalli di input superano $\{0,1\}^l$. Quindi:

$$Keys(Func(l, L)) = \{(Y_1, \ldots, Y_{2^l}) : Y_1, \ldots, Y_{2^l} \in \{0,1\}^L\}$$

è l'insieme di tutte le sequenze di lunghezza $2^l$ in cui ogni entry di una sequenza è una stringa a L bit. Per ogni $X \in \{0,1\}^l$ interpretiamo $X$ come un intero nel range $\{1, \ldots, 2^l\}$ e l'insieme:

$$Func(l, L)((Y_1, \ldots, Y_{2^l}), X) = Y_X$$

Si noti che lo spazio delle chiavi è molto grande; esso ha una dimensione pari a $2^{2^l L}$. C'è una chiave per ogni funzione da l-bit a L-bit , questo è il numero di tali funzioni. Lo spazio delle chiavi è dotato della distribuzione uniforme, tale che $f \leftarrow_R Func(l, L)$ è l'operazione di prendere una funzione random di l-bit a L-bit.

**Esempio:**

Esemplifichiamo $Func(3, 2)$, $l = 3$ e $L = 2$. Il dominio è $\{0, 1\}^3$ e il codominio è $\{0, 1\}^2$. Un esempio di istanza $f$ della famiglia di funzioni è illustrata di seguito:

| x | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| f(x) | 10 | 11 | 01 | 11 | 10 | 00 | 00 | 10 |

La chiave corrispondente a questa particolare funzione è:

$$(10, 11, 01, 11, 10, 00, 00, 10)$$

Lo spazio delle chiavi di $Func(3, 2)$ è l'insieme di tutte queste sequenze, il che significa l'insieme di tutte le tuple di 8 elementi di cui ogni componente è una stringa di due bit. Ci sono:

$$2^{2 \cdot 2^3} = 2^{16} = 65536$$

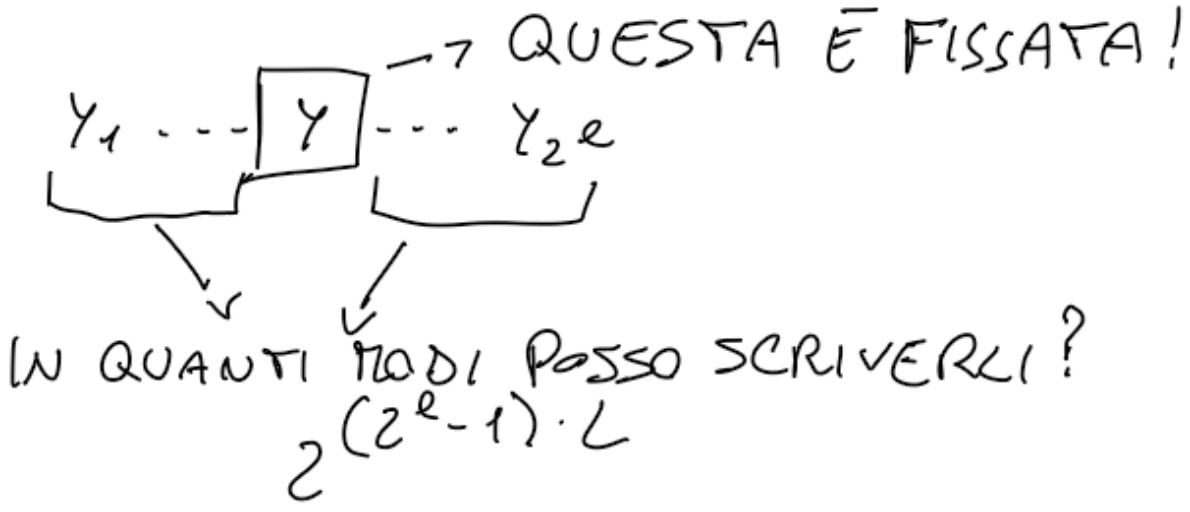
questa è la dimensione dello spazio delle chiavi.

Si noti che a input uguali corrispondono sempre output uguali. Il termine "funzione casuale" è molto fuorviante. La casualità non è riferita alla funzione ma alla scelta della stessa. Implicitamente, non abbiamo fatto altro che scegliere una funzione a caso, tra le tante possibili.

Supponiamo di conoscere la descrizione di $D$ e $R$ ed è stata già scelta la funzione $f \in Func(D, R)$, che non conosciamo. Qual è la probabilità che scegliendo $X$, $Y$ con $X \in D$ e $Y \in R$ si abbia $f(X) = Y$

$$P[f(X) = Y] = \frac{2^{(2^l - 1) \cdot L}}{2^{2^l \cdot L}} = \frac{1}{2^L}$$

Il ragionamento per ottenere questo risultato è il seguente: noi vogliamo tutte quelle funzioni che dato $X$ mi diano $Y$, quindi perdiamo L bit di libertà ovvero:



A questo punto supponiamo di sapere già che la funzione scelta nel punto $f(X_1) = Y_1$ e si vogliono trovare $X_2 \in D$ e $Y_2 \in R$ tali che:

$$Pr[f(X_2) = Y_2 \mid f(X_1) = Y_1] = \frac{2^{(2^l-2)\cdot L}}{2^{(2^l-1)\cdot L}} = \frac{1}{2^L}$$

Il fatto di conoscere $f(X_1) = Y_1$ non ci aiuta in nessun modo nel trovare una $X$ e una $Y$ con una probabilità maggiore.

I cifrari a blocchi, però, non sono delle funzioni, ma delle permutazioni.

## Permutazioni casuali

La famiglia $Perm(D)$ ha come dominio e codominio $D$. L'insieme delle istanze di $Perm(D)$ è l'insieme di tutte le permutazioni su $D$. La chiave che descrive una particolare istanza è una descrizione della funzione. In questo caso:

$$Keys(Perm(l)) = \{(Y_1, \ldots, Y_{2^l}) : Y_1, \ldots, Y_{2^l} \in \{0,1\}^l \wedge Y_1, \ldots, Y_{2^l} \text{ sono tutti diversi}\}$$

Per ogni $X \in \{0,1\}^l$ interpretiamo $X$ come un intero nel range $\{1, \ldots, 2^l\}$ e l'insieme:

$$Perm(l)((Y_1, \ldots, Y_{2^l}), X) = Y_X$$

Lo spazio delle chiavi è dotato della distribuzione uniforme, tale che $\pi \leftarrow_R Perm(l)$ è l'operazione di prendere una permutazione random su $\{0,1\}^l$. In altre parole, tutte le possibili permutazioni di $\{0,1\}^l$ sono ugualmente probabili.

**Esempio:**

Esemplifichiamo $Perm(3)$, $l = 3$. Il dominio e il codominio sono entrambi $\{0,1\}^3$. Un esempio di istanza $\pi$ della famiglia di funzioni è illustrata di seguito:

| x | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| $\pi$ | 010 | 111 | 101 | 011 | 110 | 100 | 000 | 001 |

La chiave corrispondente a questa particolare funzione è:

$$(010, 111, 101, 011, 110, 100, 000, 001)$$

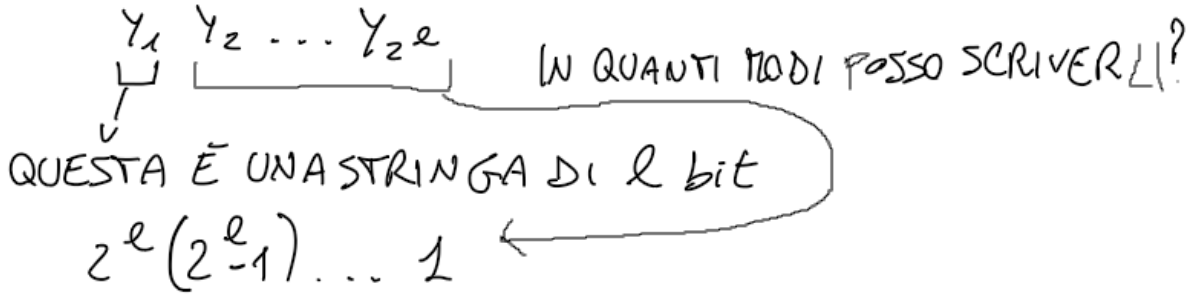
Lo spazio delle chiavi di $Perm(3)$ è l'insieme di tutte queste sequenze, il che significa l'insieme di tutte le tuple di 8 elementi di cui ogni componente è una stringa di 3 bit. Ci sono:

$$8! = 40320$$

questa è la dimensione dello spazio delle chiavi.

Supponiamo, come prima, di conoscere la descrizione di $D$ ed è stata già scelta la funzione $\pi \in Perm(D)$, che non conosciamo. Qual è la probabilità che scegliendo $X, Y$ con $X \in D$ e $Y \in D$ si abbia $\pi(X) = Y$.

$$Pr[\pi(X) = Y\ ] = \frac{(2^l - 1)!}{2^l!} = \frac{1}{2^L}$$




A questo punto supponiamo di sapere già che la funzione scelta nel punto $\pi(X_1) = Y_1$ e si vogliono trovare $X_2 \in D$ e $Y_2 \in D$ tali che:

$$Pr[\pi(X_2) = Y_2 \mid \pi(X_1) = Y_1] = \frac{(2^l - 2)!}{(2^l - 1)!} = \frac{1}{2^L - 1}$$

Il fatto di conoscere $\pi(X_1) = Y_1$ ci aiuta in qualche modo nel trovare una $X$ e una $Y$ con una probabilità maggiore.

# Funzioni pseudo-casuali

Una famiglia di funzioni che "sembrano" casuali. Una funzione pseudo-casuale è una famiglia di funzioni con la proprietà che il comportamento input-output di un'istanza casuale della famiglia è **computazionalmente indistinguibile** da quello di una funzione casuale. L'obiettivo è quello di creare delle funzioni che abbiano le proprietà delle funzioni casuali, questo perché è impossibile riuscire a trovare uno spazio di memoria in grado di contenere tutte le possibili chiavi.

Supponiamo che qualcuno abbia accesso ad una funzione attraverso una black-box, il che significa che può solo fornirgli l'input e ottenere l'output, questo soggetto ha difficoltà a dire se la funzione in questione è un'istanza casuale della famiglia di funzioni che "sembrano" casuali o una funzione casuale.

Si fissi una famiglia di funzioni $F : K \times D \to R$, dove:

- $K = \{0,1\}^k$;
- $D = \{0,1\}^l$;
- $R = \{0,1\}^L$.

Con $k, l, L \geq 1$.

Si immagini di trovarsi in una stanza con un terminale connessi a un computer fuori dalla stanza. Nel terminale è possibile scrivere qualcosa e mandarlo fuori, infine arriverà una risposta. Le domande che si possono scrivere devono essere elementi del dominio $D$, e la risposta che si riceve appartiene al codominio $R$. Il computer fuori dalla stanza implementa una funzione $g : D \to R$, tale che ogni qualvolta si scrive un valore $X$ si ottiene $g(X)$. Comunque, l'unico accesso a $g$ avviene attraverso l'interfaccia, quindi l'unica cosa che è possibile fare è inserire l'input e ottenere l'output. Si considerino due modi diversi in cui verrà scelto g, dando origine a due diversi "mondi":

- **World 0:** La funzione $g$ viene estratta a caso da $Func(D, R)$, ovvero la funzione $g$ viene selezionata tramite $g \leftarrow_R Func(D, R)$.
- **World 1:** La funzione $g$ viene estratta a caso da $F$, la funzione $g$ è selezionata tramite $g \leftarrow_R F$. (Ricorda questo significa che una chiave $k \leftarrow_R K$ e allora $g$ è settata a $F_k$).

Non viene detto quale dei due mondi è stato scelto. La scelta del mondo, e della corrispondente funzione $g$, viene effettuata prima di entrare nella stanza, cioè prima di iniziare a digitare le domande. Una volta fatte, però, queste scelte vengono fissate fino al termine della "sessione". L'obiettivo è scoprire in quale mondo ti trovi. Per fare ciò, l'unica risorsa a disposizione è il link che consente di fornire valori $X$ e recuperare $g(X)$. Dopo aver provato un certo numero di valori a scelta, bisogna prendere una decisione riguardo al mondo utilizzato. La qualità della famiglia pseudo-casuale $F$ può essere misurata dalla difficoltà di dire il mondo corretto.

Nella formalizzazione, colui che deve indovinare il mondo corretto è un algoritmo chiamato **avversario**. L'algoritmo avversario $A$ può essere randomizzato. Formalizziamo la capacità di interrogare $g$ dando ad $A$ un oracolo che prende in input qualsiasi stringa $X \in D$ e ritorna $g(X)$. Scriviamo $A^g$ per indicare che l'avversario $A$ ha accesso all'oracolo $g$. L'algoritmo $A$ può decidere quali query eseguire, magari in base alle risposte ricevute a query precedenti. Alla fine, emette un bit $b$ che è la sua decisione su quale mondo si trova. L'output del bit "1" significa che $A$ pensa di essere nel mondo 1; emettere il bit "0" significa che $A$ pensa di essere nel mondo 0. La seguente definizione associa a qualsiasi avversario di questo tipo un numero compreso tra 0 e 1 che è chiamato il suo vantaggio PRF, ed è una misura di quanto bene l'avversario sta facendo a determinare in quale mondo si trova.

**Definizione 1**

Sia $F : K \times D \to R$ una famiglia di funzioni, e sia $A$ un algoritmo che prende un oracolo per una funzione $g : D \to R$, e ritorna un bit. Consideriamo questi due esperimenti:

| $\text{Experiment } \mathbf{Exp}_F^{\text{prf-1}}(A)$ | $\text{Experiment } \mathbf{Exp}_F^{\text{prf-0}}(A)$ |
|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | $g \xleftarrow{\$} \text{Func}(D,R)$ |
| $b \xleftarrow{\$} A^{F_K}$ | $b \xleftarrow{\$} A^{g}$ |
| $\text{Return } b$ | $\text{Return } b$ |

Il vantaggio prf di $A$ è definito come segue:

$$Adv_F^{prf}(A) = |Pr[Exp_F^{prf-1}(A) = 1] - Pr[Exp_F^{prf-0}(A) = 1]|$$

Va notato che la famiglia $F$ è pubblica. L'avversario $A$, e chiunque altro, conosce la descrizione della famiglia ed è in grado, dati i valori $K, X$, di calcolare $F(K, X)$.
I *mondi* sono catturati da ciò che chiamiamo esperimenti. Il primo esperimento sceglie un'istanza casuale $F_k$ della famiglia $F$ e quindi esegue l'avversario $A$ con oracolo $g = F_K$ . L'avversario $A$ interagisce con il suo oracolo, interrogandolo e ottenendo risposte, e alla fine emette un bit. Il secondo esperimento sceglie una funzione casuale $g : D \to R$ ed esegue $A$ con questo come oracolo, restituendo nuovamente il bit di ipotesi di $A$. Ogni esperimento ha una certa probabilità di restituire 1. La probabilità viene sostituita dalle scelte casuali fatte nell'esperimento. Quindi, per il primo esperimento, la probabilità si basa sulla scelta di $K$ e di qualsiasi scelta casuale che $A$ potrebbe fare, poiché $A$ può essere un algoritmo randomizzato. Nel secondo esperimento, la probabilità si basa sulla scelta casuale di $g$ e su qualsiasi scelta casuale fatta da $A$. Queste due probabilità dovrebbero essere valutate separatamente; i due esperimenti sono completamente diversi.
Per vedere quanto bene fa $A$ nel determinare in quale mondo si trova, osserviamo la differenza nelle probabilità che i due esperimenti restituiscano 1. Se $A$ sta facendo un buon lavoro nel dire in quale mondo si trova, restituirà 1 più spesso nel primo esperimento che nel secondo. Quindi la differenza è una misura di quanto bene sta andando $A$. Chiamiamo questa misura il **vantaggio**

**prf** di $A$. La si pensi come la probabilità che $A$ "break" lo schema $F$ , con "break" interpretato in un modo specifico e tecnico basato sulla definizione 1.

Si noti che la definizione 1 non limita il comportamento dell'avversario. Possiamo però limitare, il numero di domande che è autorizzato a porre, il suo tempo di calcolo, la lunghezza totale delle domande. La “sicurezza” della famiglia $F$ come funzione pseudo-casuale va quindi pensata come dipendente dalle risorse concesse all'aggressore. Potremmo voler sapere, per ogni data limitazione delle risorse, qual è il vantaggio prf ottenuto dall'avversario più "intelligente" tra tutti coloro che sono limitati ai limiti delle risorse date.

La definizione 1 di limita ad associare un certo vantaggio. Che vuol dire allora che una prf $F$ è sicura?
La definizione 1 non ha alcuna condizione o affermazione esplicita riguardo a quando $F$ dovrebbe essere considerata *sicura*. Associa solo a qualsiasi avversario $A$ che attacca $F$ una funzione **vantaggio prf**. Intuitivamente, $F$ è *sicura* se il valore della funzione di vantaggio è "basso" per tutti gli avversari le cui risorse sono "pratiche".

Riassumiamo le caratteristiche principali del quadro di definizione 1 come le vedremo emergere in seguito. In primo luogo, ci sono esperimenti, che coinvolgono un avversario. Esiste poi una funzione vantaggio associata ad un avversario che restituisce la probabilità che l'avversario in questione “rompi” lo schema. Queste due componenti saranno presenti in tutte le definizioni. Ciò che varia sono gli esperimenti; qui è dove definiamo come misuriamo la sicurezza.

**Esempio:**

Sia $F : K \times D \to R$ una famiglia di funzioni e i tre insiemi definiti nel seguente modo:

- $K = \{0,1\}^k$
- $D = \{0,1\}^l$
- $R = \{0,1\}^L$

Inoltre, $k = l \cdot L$.

Una chiave è definita nel seguente modo:

$$\begin{bmatrix} k[1,1] & \dots & k[1,l] \\ . & . & . \\ . & . & . \\ . & . & . \\ k[L,1] & \dots & k[L,l] \end{bmatrix}$$

Il valore $F_k(x)$ è definito nella seguente maniera:

$$F_k(x) = \begin{bmatrix} k[1,1] & \dots & k[1,l] \\ . & . & . \\ . & . & . \\ . & . & . \\ k[L,1] & \dots & k[L,l] \end{bmatrix} * \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_l \end{bmatrix} = \begin{bmatrix} y_1 \\ . \\ . \\ . \\ y_L \end{bmatrix}$$

Il valore $y_n$ è un bit e si calcola nel seguente modo:

$$y_n = (k[i,1] \wedge x_1) \vee \dots \vee (k[i,l] \wedge x_l)$$

Questa famiglia è pseudo-casuale?

Bisogna trovare un avversario per cui il vantaggio non è piccolo.

Supponiamo che l'avversario scelga $x = 0^l$, allora:

$$F_k(0^l) = 0^L \ \forall k \in K$$

Supponiamo che l'avversario sia il seguente algoritmo:

```
A(F) {
    x <- 0^l
    y <- O_PRF(x)
    if(y == 0^L) return 1
    else return 0
}
```

Calcoliamo il vantaggio dell'avversario:

- Caso prf-1:
  - $$Pr[Esp_F^{prf-1}(A) = 1] = 1$$
  - La probabilità è sempre 1 quando $x$ vale zero.
- Caso prf-0:
  - $$Pr[Esp_F^{prf-0}(A) = 1] = \frac{1}{2^L}$$
  - In questo caso quando si utilizza una funzione casuale e la probabilità che l'avversario riesca ad indovinare dipende dal modo in cui le funzioni casuali mappano l'output.

Quindi il vantaggio dell'avversario è:

$$Adv(A) = 1 - \frac{1}{2^L} \simeq 1$$

Quindi la famiglia di funzioni appena analizzata non è prf sicura.

**Esempio 2:**

Sia $F : K \times D \to R$ una famiglia di funzioni, tale che $K = D = R = \{0,1\}^n$ e $F_k(x) = y = (x \oplus k)$.

Supponiamo che l'avversario sia il seguente algoritmo:

```
A(F) {
    x_1 <-_R D
    y_1 <- (x_1 XOR y_1)
    x_2 <- O_PRF(X_2)
    if(y_2 == 0^n) return 1
    else return 0
}
```

Calcoliamo il vantaggio dell'avversario:

- Caso prf-1:
  - $$Pr[Esp_F^{prf-1}(A) = 1] = 1$$
  - Perché $x_1 \oplus y_1 = x_1 \oplus x_1 \oplus k = k = x_2$, quindi $x_2 \oplus k = k \oplus k = 0$
- Caso prf-0:

- $$Pr[Esp_F^{prf-0}(A) = 1] = \frac{1}{2^n}$$
  - In questo caso quando si utilizza una funzione casuale e la probabilità che l'avversario riesca ad indovinare dipende dal modo in cui le funzioni casuali mappano l'output.

Creare una PRF è difficile, perché basta un input debole per rompere lo schema.

Lo stesso ragionamento visto fino adesso può essere adottato per le permutazioni, ma una differenza è legata al fatto che l'utente può fare altre tipologie di domande.

# Permutazioni pseudo-casuali

Una famiglia di permutazioni $F : K \times D \to D$ è una permutazione pseudo-casuale se il comportamento input-output di un'istanza casuale della famiglia è "computazionalmente indistinguibile" da quello di una permutazione casuale su $D$.

In questa impostazione, ci sono due tipi di attacchi che si possono prendere in considerazione. Uno, come prima, è che l'avversario ottiene un oracolo per la funzione $g$ che viene testata. Tuttavia, quando $F$ è una famiglia di permutazioni, si può anche considerare il caso in cui l'avversario ottiene, in aggiunta, un oracolo per $g^{-1}$. Consideriamo queste impostazioni a turno. Il primo è l'impostazione degli attacchi con testo in chiaro scelto (**chosen plaintext attack**), mentre il secondo è l'impostazione degli attacchi con testo cifrato scelto (**chosen ciphertext attack**).

## PRP sotto CPA

Fissiamo una famiglia di permutazioni $F : K \times D \to D$. ($K = \{0,1\}^k$ e $D = \{0,1\}^l$ , poiché questo è il caso più comune). Come prima, consideriamo un avversario $A$ che si trova in una stanza dove ha accesso oracolare a una funzione $g$ scelta in uno dei due modi.

- **World 0:** La funzione $g$ viene estratta a caso da $Perm(D)$, ovvero la funzione $g$ viene selezionata tramite $g \leftarrow_R Perm(D)$.
- **World 1:** La funzione $g$ viene estratta a caso da $F$, la funzione $g$ è selezionata tramite $g \leftarrow_R F$. (Ricorda questo significa che una chiave $k \leftarrow_R K$ e allora $g$ è settata a $F_k$).

**Definizione 2**

Sia $F : K \times D \to D$ una famiglia di permutazioni, e sia $A$ un algoritmo che usa un oracolo per una funzione $g : D \to D$, e ritorna un bit. Consideriamo due esperimenti:

| $\text{Experiment } \mathbf{Exp}_F^{\text{prp-cpa-1}}(A)$ | $\text{Experiment } \mathbf{Exp}_F^{\text{prp-cpa-0}}(A)$ |
|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | $g \xleftarrow{\$} \text{Perm}(D)$ |
| $b \xleftarrow{\$} A^{F_K}$ | $b \xleftarrow{\$} A^{g}$ |
| $\text{Return } b$ | $\text{Return } b$ |

Il **vantaggio cpa prp** di $A$ è definito come segue:

$$Adv_F^{prp-cpa}(A) = |Pr[Exp_F^{prp-cpa-1}(A) = 1] - Pr[Exp_F^{prp-cpa-0}(A) = 1]|$$

L'intuizione è simile a quella della Definizione 1. La differenza è che qui l'oggetto "ideale" con cui $F$ viene confrontato non è più la famiglia delle funzioni casuali, ma piuttosto la famiglia delle permutazioni casuali. Anche le convenzioni relative alle misure sulle risorse rimangono le stesse di prima. Informalmente, una famiglia $F$ è una PRP sicura sotto CPA se il vantaggio dell'avversario è piccolo per tutti gli avversari che utilizzano una quantità pratica di risorse.

## PRP sotto CCA

Fissiamo una famiglia di permutazioni $F : K \times D \to D$. ($K = \{0,1\}^k$ e $D = \{0,1\}^l$ , poiché questo è il caso più comune). Come prima, consideriamo un avversario $A$ che si trova in una stanza, ma ora ha accesso oracolare a due funzioni, $g$ e la sua funzione inversa $g^{-1}$. Il modo in cui viene scelta $g$ è lo stesso del caso CPA, e una volta scelta $g$, $g^{-1}$ viene automaticamente definita, quindi non dobbiamo dire come viene scelta.

- **World 0:** La funzione $g$ viene estratta a caso da $Perm(D)$, ovvero la funzione $g$ viene selezionata tramite $g \leftarrow_R Perm(D)$.
- **World 1:** La funzione $g$ viene estratta a caso da $F$, la funzione $g$ è selezionata tramite $g \leftarrow_R F$. (Ricorda questo significa che una chiave $k \leftarrow_R K$ e allora $g$ è settata a $F_k$).

In World 1 sia $g^{-1} = F_k^{-1}$ l'inverso dell'istanza scelta, mentre in World 0 è l'inverso della permutazione casuale scelta. Come prima, il compito che deve affrontare l'avversario $A$ è determinare in quale mondo è stato collocato in base al comportamento input-output dei suoi oracoli.

**Definizione 3**

Sia $F : K \times D \to D$ una famiglia di permutazioni, e sia $A$ un algoritmo che usa un oracolo per una funzione $g : D \to D$, e inoltre un oracolo per una funzione $g^{-1} : D \to D$, e ritorna un bit. Consideriamo due esperimenti:

| $\text{Experiment } \mathbf{Exp}_F^{\text{prp-cca-1}}(A)$ | $\text{Experiment } \mathbf{Exp}_F^{\text{prp-cca-0}}(A)$ |
|---|---|
| $K \xleftarrow{\$} \mathcal{K}$ | $g \xleftarrow{\$} \text{Perm}(D)$ |
| $b \xleftarrow{\$} A^{F_K, F_K^{-1}}$ | $b \xleftarrow{\$} A^{g, g^{-1}}$ |
| $\text{Return } b$ | $\text{Return } b$ |

Il **vantaggio prp cca** di $A$ è definito come:

$$Adv_F^{prp-cca} = |Pr[Exp_F^{prp-cca-1}(A) = 1] - Pr[Exp_F^{prp-cca-0}(A) = 1]|$$

L'intuizione è simile a quella della Definizione 1. La differenza è che qui l'avversario ha più potere: non solo può interrogare $g$, ma può interrogare direttamente $g^{-1}$. Anche le convenzioni relative alle misure sulle risorse rimangono le stesse di prima. Tuttavia, saremo interessati ad alcuni parametri aggiuntivi delle risorse. In particolare, poiché ora ci sono due oracoli, possiamo contare separatamente il numero di query e la lunghezza totale di queste query per ciascuno. Come al solito, informalmente, una famiglia $F$ è una PRP sicura sotto CCA se il vantaggio è piccolo per tutti gli avversari che utilizzano una quantità pratica di risorse.

## Relazione tra i due attacchi

Se un avversario non interroga $g^{-1}$, l'oracolo potrebbe anche non essere lì, e l'avversario sta effettivamente organizzando una CPA. Quindi si ha quanto segue:

**Preposizione**

Sia $F : K \times D \to D$ una famiglia di permutazioni e sia $A$ un avversario (attacco PRP-CPA). Si supponga che $A$ impieghi un tempo $t$ per essere eseguito, chieda $q$ query e queste totalizzano $\mu$ bit. Quindi esiste un avversario $B$ (attacco PRP-CCA) che impiega un tempo $t$ per essere eseguito, chiede $q$ query che totalizzano $\mu$ bit e non richiede query di testo cifrato scelto, dove:

$$Adv_F^{prp-cca}(B) \geq Adv_F^{prp-cpa}(A)$$

Sebbene il risultato tecnico sia facile, vale la pena fare un passo indietro per spiegarne l'interpretazione. La preposizione dice che se hai un avversario $A$ che rompe $F$ nel senso PRP-CPA, allora hai qualche altro avversario $B$ che rompe $F$ nel senso PRP-CCA. Inoltre, l'avversario $B$ sarà efficiente quanto lo era l'avversario $A$. Di conseguenza, se pensi che non ci sia un avversario ragionevole $B$ che rompa $F$ nel senso PRP-CCA, allora non hai altra scelta che credere che non ci sia un avversario ragionevole $A$ che rompa $F$ nel senso PRP-CPA. L'inesistenza di un avversario ragionevole $B$ che rompe $F$ nel senso PRP-CCA significa che $F$ è PRP-CCA sicuro, mentre l'inesistenza di un avversario ragionevole $A$ che rompe $F$ nel senso PRP-CPA significa che $F$ è sicuro PRP-CPA . Quindi la sicurezza PRP-CCA implica la sicurezza PRP-CPA.

# Applicazione ai cifrari a blocchi

Una delle motivazioni principali per le nozioni di funzioni pseudo-casuali (PRF) e permutazioni pseudo-casuali (PRP) è modellare i cifrari a blocchi e quindi consentire l'analisi della sicurezza dei protocolli che utilizzano i cifrari a blocchi.

Come discusso nel capitolo sui cifrari a blocchi, classicamente la sicurezza di DES o altri cifrari a blocchi è stata esaminata solo per quanto riguarda il recupero delle chiavi. Cioè, l'analisi di un codice a blocchi $F$ si è concentrata sulla seguente domanda: dato un certo numero di esempi input-output

$$(X_1, F_k(X_1)), \ldots, (X_q, F_k(X_q))$$

dove $K$ è una chiave casuale non conosciuta, quanto è difficile trovare $K$?

Il cifrario a blocchi è considerato “sicuro” se le risorse necessarie per recuperare la chiave sono proibitive. Tuttavia, come abbiamo visto, la durezza della **key recovery** non è sufficiente per la sicurezza. Avevamo discusso di volere una proprietà di sicurezza principale dei cifrari a blocchi in base alla quale gli usi naturali dei cifrari a blocchi potessero essere dimostrati sicuri. Suggeriamo che questa proprietà principale sia che il cifrario a blocchi sia una PRP sicura, sotto CPA o CCA. Non possiamo dimostrare che specifici cifrari a blocchi abbiano questa proprietà. Il meglio che possiamo fare è presumere che lo facciano e poi continuare a usarli. Per le valutazioni di sicurezza quantitative, faremmo congetture specifiche sulle funzioni di vantaggio di vari cifrari a blocchi. Potrebbero esistere attacchi altamente efficaci che rompono DES o AES come PRF senza recuperare la chiave. Finora, non sappiamo di tali attacchi, ma la qualità dello sforzo critto-analitico che si è concentrato su questo obiettivo è piccola. Certamente, presumere che un cifrario a blocchi sia una PRF è un presupposto molto più forte di quello che è la sicurezza in senso key-recovery. Tuttavia, la motivazione e gli argomenti che abbiamo delineato a favore dell'assunzione della PRF rimangono, e la nostra opinione è che se un codice a blocchi viene violato come PRF, allora dovrebbe essere considerato insicuro e dovrebbe essere creato un sostituto.

# Sicurezza contro Key Recovery

Abbiamo menzionato più volte che la sicurezza contro il recupero delle chiavi non è sufficiente come nozione di sicurezza per un cifrario a blocchi. Tuttavia è certamente necessario: se il recuperodella chiave è facile, il codice a blocchi dovrebbe essere dichiarato non sicuro. Abbiamo indicato che vogliamo adottare come nozione di sicurezza per un cifrario a blocchi la nozione di PRF o PRP. Se questo deve essere fattibile, dovrebbe essere il caso che qualsiasi famiglia di funzioni non sicura durante il recuperodella chiave sia insicura anche come PRF o PRP. In questa sezione verifichiamo questo semplice fatto. Ciò ci consentirà di esercitare il metodo delle **riduzioni**.
Iniziamo formalizzando la sicurezza contro il recupero delle chiavi. Consideriamo un avversario che, sulla base di esempi input-output di un'istanza $F_k$ della famiglia $F$ , cerca di trovare $k$. Il suo vantaggio è definito come la probabilità che riesca a trovare $k$. La probabilità è superiore alla scelta casuale di $k$, e eventuali scelte casuali dell'avversario stesso.
Diamo all'oracolo avversario l'accesso a $F_k$ in modo che possa ottenere esempi di input-output a sua scelta. Non limitiamo l'avversario riguardo al metodo che usa. Questo porta alla seguente definizione:

**Definizione 4**

Sia $F : K \times D \rightarrow R$ una famiglia di funzioni, e sia $B$ un algoritmo che usa un oracolo per la funzione $g : D \rightarrow R$ e ritorna in output una stringa. Consideriamo il seguente esperimento:

$$\begin{aligned}&\text{Experiment } \mathbf{Exp}_F^{\text{kr}}(B)\\ &\quad K \xleftarrow{\$} \mathsf{Keys}(F)\\ &\quad K' \leftarrow B^{F_K}\\ &\quad \text{If } K = K' \text{ then return 1 else return 0}\end{aligned}$$

Il **vantaggio kr** di $B$ è definito come:

$$Adv_F^{kr}(B) = Pr[Exp_F^{kr}(B) = 1] \simeq \frac{1}{|K|}$$

Questa definizione è stata resa abbastanza generale in modo da catturare tutti i tipi di attacchi di recuperodelle chiavi. Tutti gli attacchi classici come la ricerca esauriente della chiave, la crittoanalisi differenziale o la crittoanalisi lineare corrispondono a scelte diverse e specifiche dell'avversario $B$. Rientrano in questo quadro perché tutti hanno l'obiettivo di trovare la chiave $k$ sulla base di un certo numero di esempi input-output di un'istanza $F_k$ del cifrario.

**Preposizione**

Sia $F : K \times D \rightarrow R$ una famiglia di funzioni, e sia $B$ un avversario di recupero chiave contro $F$ . Supponiamo che il tempo di esecuzione di $B$ sia al massimo $t$ e produca al massimo $q < |D|$ interrogazioni oracolari. Allora esiste un avversario PRF $A$ contro $F$ tale che $A$ ha un tempo di esecuzione al massimo $t$ più il tempo per un calcolo di $F$, fa al massimo $q + 1$ query oracolari e si ha che:

$$Adv_F^{kr}(B) \leq Adv_F^{prf}(A) + \frac{1}{|R|}$$

La preposizione dimostra che se $F$ è una famiglia di funzioni sicura in senso PRF allora $F$ è sicura contro ogni avversario (limitato) di tipo key recovery.

Il problema che l'avversario $A$ sta cercando di risolvere è determinare se il suo dato oracolo $g$ è un'istanza casuale di $F$ o una funzione casuale di $D$ in $R$. $A$ eseguirà $B$ come subroutine e utilizzerà l'output di $B$ per risolvere il proprio problema.

$B$ è un algoritmo che sfrutta un oracolo $F_k$ per una chiave casuale $k \in K$ e cerca di trovare $k$ facendo delle richieste al suo oracolo. Per semplicità, supponiamo che $B$ non effettui nessuna richiesta al suo oracolo. Ora, quando $A$ esegue $B$, produce una chiave $k'$. $A$ può testare $k'$ controllando se $F(k', x)$ concorda con $g(x)$ per qualche valore di $x$. Se è così, scommette che $g$ era un'istanza di $F$, in caso contrario scommette che $g$ era casuale.

Se $B$ effettua una richiesta all'oracolo, dobbiamo chiederci come $A$ può eseguire $B$. L'oracolo che $B$ vuole non è disponibile a livello di codice, ovvero l'avversario non ha accesso al sorgente dell'oracolo, è come se usasse una funzione di libreria. Tuttavia, $B$ è un pezzo di codice, che comunica con il suo oracolo tramite un'interfaccia prescritta. Se si inizia ad eseguire $B$, a un certo punto si produrrà una richiesta all'oracolo e $B$ attende una risposta. Quando quest'ultima appare, $B$ continua la sua esecuzione. Al termine delle richieste all'oracolo, $B$ restituirà il suo output. Ora, quando $A$ esegue $B$, fornirà esso stesso  le risposte alle domande dell'oracolo di $B$. Quando $B$ si ferma, dopo aver fatto qualche domanda, $A$ inserirà la risposta nella locazione di memoria prescritta e lascerà $B$ libero di continuare la sua esecuzione. $B$ non conosce la differenza tra questo oracolo "simulato" e l'oracolo reale se non nella misura in cui può ricavarlo dai valori restituiti.

Il valore che $B$ si aspetta in risposta alla richiesta $x$ è $F_k(x)$ dove $k$ è una chiave scelta in maniera casuale dall'insieme $K$. Tuttavia, $A$ ritorna ad esso come risposta per la richiesta $x$ il valore $g(x)$, dove $g$ è l'oracolo di $A$. Quando $g(x)$ è un'istanza di $F$ e quindi $B$ funziona come farebbe nel suo ambiente abituale, restituirà la chiave $k$ con una probabilità pari al suo vantaggio kr. Tuttavia, quando $g$ è una funzione casuale, $B$ restituisce valori che hanno poca relazione con quelli che ci si aspetta, in questo caso non si ha idea di quali proprietà abbia l'output. La chiave generata da $B$ viene utilizzata da $A$ per assicurarsi che sia la chiave corretta, utilizzando la sua scelta di $x$.

In breve, l'avversario $A$ deve determinare se una data funzione $g$ appartiene alla famiglia $F$ o è una funzione casuale da $D$ a $R$. $A$ ha la possibilità di "sfruttare" l'altro avversario $B$.

**Dimostrazione della preposizione**

All'avversario $A$ verrà fornito un oracolo per una funzione $g : D \to R$, e cercherà di determinare in quale *mondo* si trova. Per fare ciò, eseguirà l'avversario $B$ come una subroutine. Forniamo la descrizione seguita da una spiegazione e analisi.

*Avversario A:*

$i \leftarrow 0$
Run adversary $B$, replying to its oracle queries as follows
When $B$ makes an oracle query $x$ do
    $i \leftarrow i + 1$ ; $x_i \leftarrow x$
    $y_i \leftarrow g(x_i)$
    Return $y_i$ to $B$ as the answer
Until $B$ stops and outputs a key $K'$
Let $x$ be some point in $D - \{x_1, \ldots, x_q\}$
$y \leftarrow g(x)$
**if** $F(K', x) = y$ **then return** 1 **else return** 0

Possiamo osservare che $A$ sta eseguendo $B$ e fornisce esso stesso le risposte alle domande dell'oracolo di $B$ tramite l'oracolo $g$. Quando $B$ è stato completato, restituisce una chiave $k' \in K$, che $A$ verifica se $F(k', x)$ concorda con $g(x)$. Qui $x$ è un valore diverso da quello interrogato da $B$, ed è per garantire che tale valore possa essere trovato in un numero di richieste $q < |D|$. Adesso supponiamo che:

$$Pr[Exp_F^{prf-1}(A) = 1] \geq Adv_F^{kr}(B) \qquad (1)$$

$$Pr[Exp_F^{prf-0}(A) = 1] = \frac{1}{|R|} \qquad (2)$$

Stabiliamo il vantaggio di $A$ come segue:

$$Adv_F^{prf}(A) = Pr[Exp_F^{prf-1} = 1] - Pr[Exp_F^{prf-0}(A) = 1] \geq Adv_F^{kr}(B) - \frac{1}{|R|}$$

L'equazione (1) è vera perché l'oracolo $g$ in $Exp_F^{prf-1}(A)$ è un'istanza casuale di $F$, che è l'oracolo che $B$ si aspetta, e quindi $B$ funziona come in $Exp_F^{kr}(B)$. Se $B$ ha successo, il che significa che la chiave $k'$ che emette è uguale a $k$, allora certamente $A$ restituisce 1.

E' possibile che $A$ possa restituire 1 anche se $B$ non ha avuto successo. Ciò accadrebbe se $k' \neq k$ ma $F(k', x) = F(k, x)$. E' per questo motivo che l'equazione (1) è in una disuguaglianza piuttosto che in un'uguaglianza.

L'equazione (2) è vera perché in $Exp_F^{prf-0}(A)$ la funzione $g$ è casuale e poiché $x$ non è mai stata interrogata da $B$, il valore $g(x)$ è imprevedibile per $B$. Immagina che $g(x)$ sia scelto solo quando $x$ viene interrogato su $g$. A quel punto, $k'$, e quindi $F(k', x)$, è già definito. Quindi $g(x)$ ha $\frac{1}{|R|}$ possibilità di colpire questo punto fisso. Questo è vero indipendentemente da quanto $B$ tenti di rendere $F(K', x)$ uguale a $g(x)$.