Dallo spettro alla sicurezza: Expander Graphs tra teoria e crittografia

Daniele Muscillo

Mentore: Daniele Venturi

Fondazione ELICSIR

Bertinoro, 21 marzo 2026

Roadmap

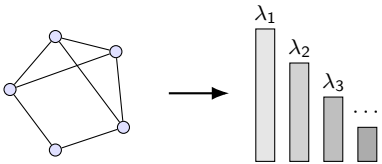
- **Expanders & Spectral Graph Theory**:
  cosa è un *expander* e cosa "misura" lo spettro

Lo spettro come lente

Roadmap

- **Expanders & Spectral Graph Theory**:
  cosa è un *expander* e cosa "misura" lo spettro

Lo spettro come lente



gli autovalori catturano proprietà globali, come connettività e l'essere un expander o meno

Roadmap

- **Expanders & Spectral Graph Theory**: cosa è un *expander* e cosa "misura" lo spettro

Dallo spettro al mixing

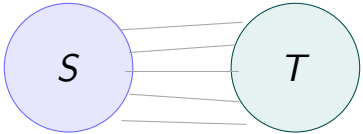
dallo spettro a contare quanti archi cadono tra due insiemi

Roadmap

- **Expanders & Spectral Graph Theory**:
  cosa è un *expander* e cosa "misura" lo spettro
- **Expander Mixing Lemma**:
  una misura di randomicità del grafo

Dallo spettro al mixing

dallo spettro a contare quanti archi cadono tra due insiemi



$$|E(S,T)| \approx \frac{d}{n}|S||T|$$

il mixing lemma quantifica quanto il grafo si comporti come uno casuale

Roadmap

- **Expanders & Spectral Graph Theory**:
  cosa è un *expander* e cosa "misura" lo spettro
- **Expander Mixing Lemma**:
  una misura di randomicità del grafo
- **Robust Secret Sharing**:
  una applicazione in crittografia (2016)

Mantenere un segreto con impostori

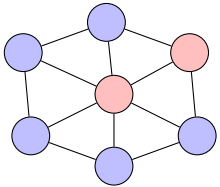
condividere un segreto con dei giocatori, in modo che possano ricostruire il segreto solo collaborando

Roadmap

- **Expanders & Spectral Graph Theory**:
  cosa è un *expander* e cosa "misura" lo spettro
- **Expander Mixing Lemma**:
  una misura di randomicità del grafo
- **Robust Secret Sharing**:
  una applicazione in crittografia (2016)

Mantenere un segreto con impostori

condividere un segreto con dei giocatori, in modo che possano ricostruire il segreto solo collaborando



e anche se degli impostori cercano di ostcolarla

Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

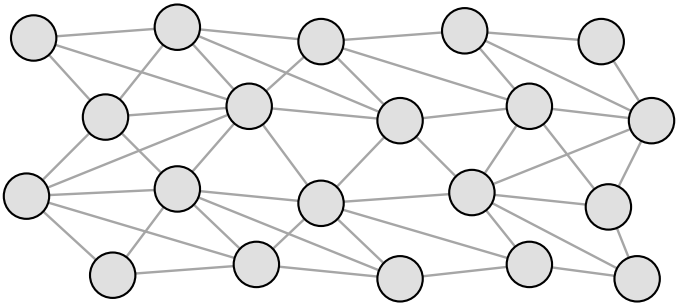
$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$

Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$

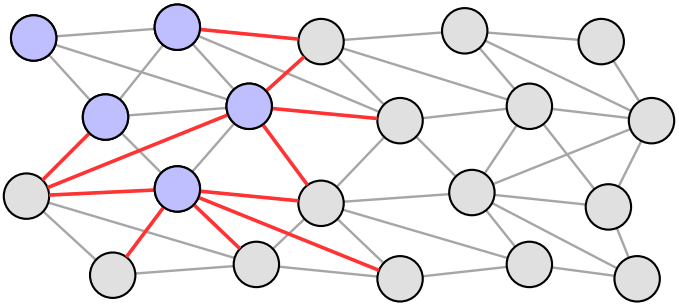


Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$

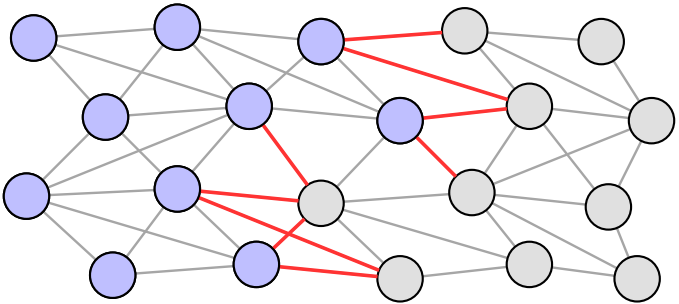


Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$

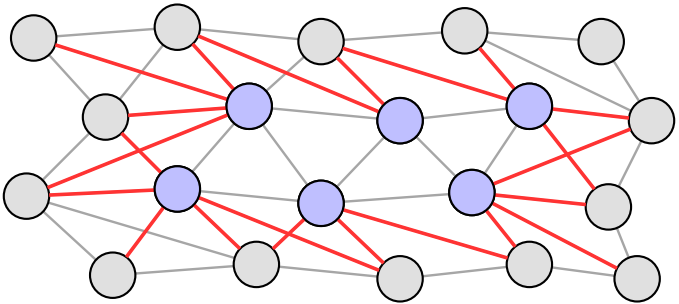


Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$

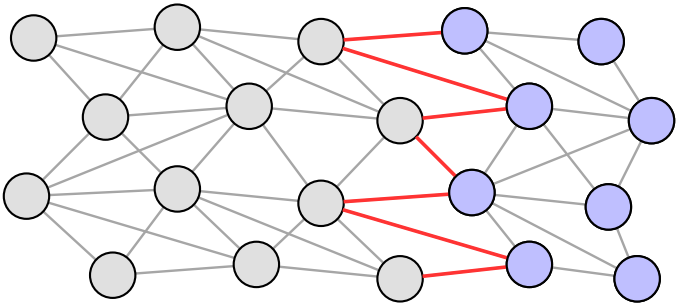


Expander Graphs: una definizione combinatoria

The Cheeger constant

Sia $G = (V, E)$ un grafo non orientato, la sua *costante di espansione*

$$h(G) := \min_{0<|S|\leq|V|/2} \frac{|E(S, V \setminus S)|}{|S|}.$$



Dal taglio ottimo...

La costante $h(G)$ è la soluzione di un complesso problema di **ottimizzazione combinatoria**.

Dal taglio ottimo...

La costante $h(G)$ è la soluzione di un complesso problema di **ottimizzazione combinatoria**.

Ostacolo Computazionale

Calcolare il taglio ottimo in modo esatto è **NP-hard**.

Dal taglio ottimo...

La costante $h(G)$ è la soluzione di un complesso problema di **ottimizzazione combinatoria**.

Ostacolo Computazionale

Calcolare il taglio ottimo in modo esatto è **NP-hard**.

Ma come per molti problemi intrattabili...

Dal taglio ottimo... alla matrice di adiacenza

La costante $h(G)$ è la soluzione di un complesso problema di **ottimizzazione combinatoria**.

Ostacolo Computazionale

Calcolare il taglio ottimo in modo esatto è **NP-hard**.

Ma come per molti problemi intrattabili...

Un'approssimazione algebrica

Possiamo ottenere **stime molto accurate** studiando un oggetto puramente algebrico:
lo **spettro della matrice di adiacenza** A.

Grafi non diretti d-regolari: conseguenze

Il grafo G e la sua matrice di adiacenza

Sia $G = (V, E)$ un grafo non orientato d-**regolare**. La sua matrice di adiacenza A è:

$$A_{uv} = 1 \iff (u, v) \in E$$

Grafi non diretti d-regolari: conseguenze

Il grafo G e la sua matrice di adiacenza

Sia $G = (V, E)$ un grafo non orientato d-**regolare**. La sua matrice di adiacenza A è:

$$A_{uv} = 1 \iff (u, v) \in E$$

- **Simmetria**
  La matrice A è simmetrica, ossia $A = A^T$, e i suoi autovalori sono reali:
  $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$

Grafi non diretti d-regolari: conseguenze

Il grafo G e la sua matrice di adiacenza

Sia $G = (V, E)$ un grafo non orientato d-**regolare**. La sua matrice di adiacenza A è:

$$A_{uv} = 1 \iff (u, v) \in E$$

- **Simmetria**
  La matrice A è simmetrica, ossia $A = A^T$, e i suoi autovalori sono reali:
  $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$

- **Limitazione**
  Il primo autovalore è sempre uguale, tutti gli altri sono contenuti nell'intervallo $[-d, d]$

Grafi non diretti d-regolari: conseguenze

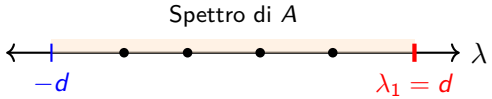
Il grafo G e la sua matrice di adiacenza

Sia $G = (V, E)$ un grafo non orientato d-**regolare**. La sua matrice di adiacenza A è:

$$A_{uv} = 1 \iff (u, v) \in E$$

- **Simmetria**
  La matrice A è simmetrica, ossia $A = A^T$, e i suoi autovalori sono reali:
  $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$

- **Limitazione**
  Il primo autovalore è sempre uguale, tutti gli altri sono contenuti nell'intervallo $[-d, d]$



Gli autovalori di A come soluzioni di problemi sul grafo

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

Gli autovalori di A come soluzioni di problemi sul grafo

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(1) Connettività

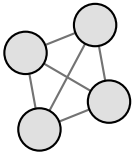
La molteplicità di $\lambda_i(A) = d$ è uguale al **numero di componenti connesse** di G.

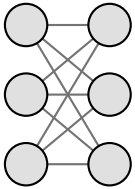
Gli autovalori di A come soluzioni di problemi sul grafo

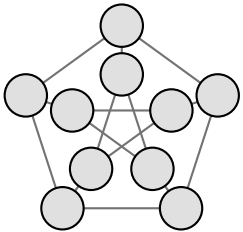
Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(1) Connettività

La molteplicità di $\lambda_i(A) = d$ è uguale al **numero di componenti connesse** di G.





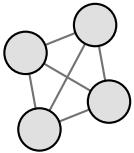


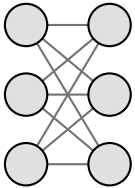
Gli autovalori di A come soluzioni di problemi sul grafo

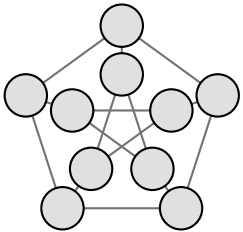
Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(1) Connettività

La molteplicità di $\lambda_i(A) = d$ è uguale al **numero di componenti connesse** di G.







$$\lambda_1(A) = d$$
$$\lambda_2(A) = d$$
$$\lambda_3(A) = d$$
$$\lambda_4(A) < d$$
$$\vdots$$

Gli autovalori di A come soluzioni di problemi sul grafo

Spettro di A : $$d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$$

(2) Cheeger's inequality

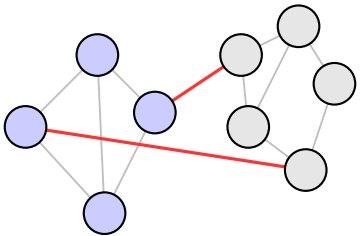
$$\frac{d - \lambda_2(A)}{2} \leq h(G) := \min_{|S| \leq \frac{|V|}{2}} \frac{|E(S, V \setminus S)|}{|S|} \leq \sqrt{2d(d - \lambda_2(A))}.$$

Gli autovalori di A come soluzioni di problemi sul grafo

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(2) Cheeger's inequality

$$\frac{d - \lambda_2(A)}{2} \leq h(G) := \min_{|S| \leq \frac{|V|}{2}} \frac{|E(S, V \setminus S)|}{|S|} \leq \sqrt{2d(d - \lambda_2(A))}.$$



Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d|S||T|}{n} \right| \leq \sigma_2(A)\sqrt{|S||T|}$$

Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.

Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d|S||T|}{n} \right| \leq \sigma_2(A)\sqrt{|S||T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.

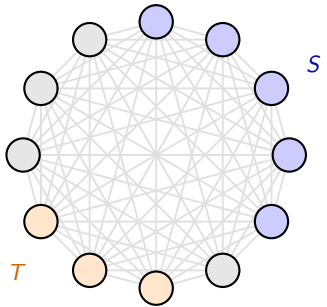
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



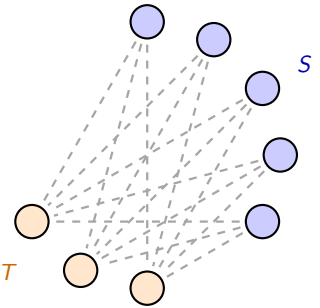
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



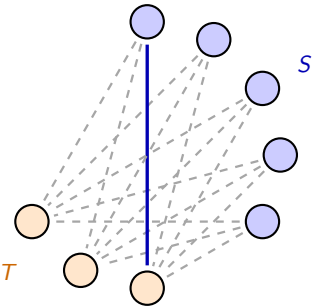
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



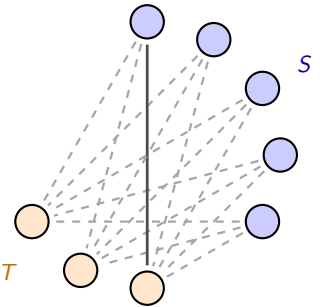
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



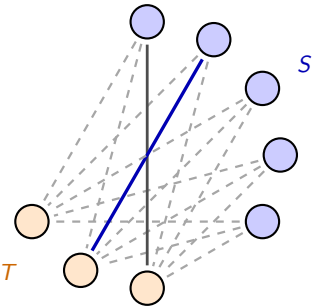
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



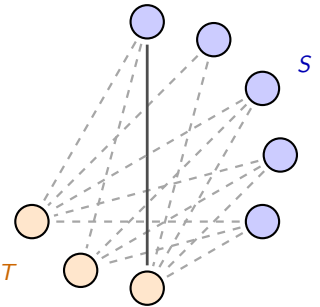
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d|S||T|}{n} \right| \leq \sigma_2(A)\sqrt{|S||T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



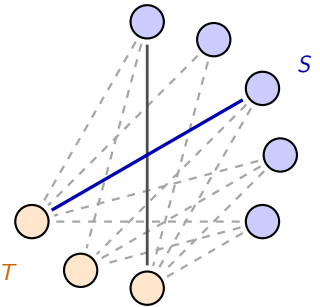
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



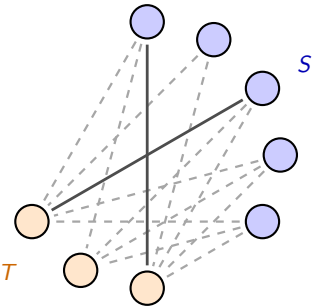
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d|S||T|}{n} \right| \leq \sigma_2(A)\sqrt{|S||T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



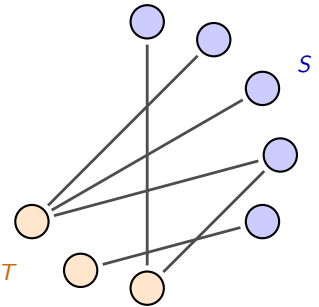
Una misura di randomicità del grafo

Expander Mixing Lemma

Sia G un grafo d-regolare su n vertici, sia $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ lo spettro associato alla matrice di adiacenza A. Sia $\sigma_2(A) := \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$. Allora

$$\forall S, T \subset V(G) \text{ si ha che } \left| E(S,T) - \frac{d\,|S|\,|T|}{n} \right| \leq \sigma_2(A)\sqrt{|S|\,|T|}$$

- $E(S,T)$: numero di archi con un estremo in S e l'altro in T.
- $\frac{d|S||T|}{n}$: il **valore atteso** del numero di archi tra S e T in un grafo casuale in cui ogni nodo ha grado medio d.



Dallo spettro alla crittografia

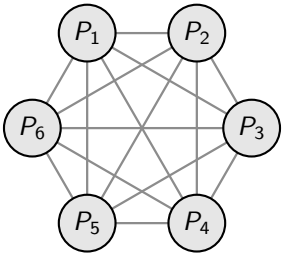
Robust Secret Sharing

Il mixing quasi casuale degli expander diventa robustezza contro avversari.

t-Secret Sharing: obiettivo e definizione

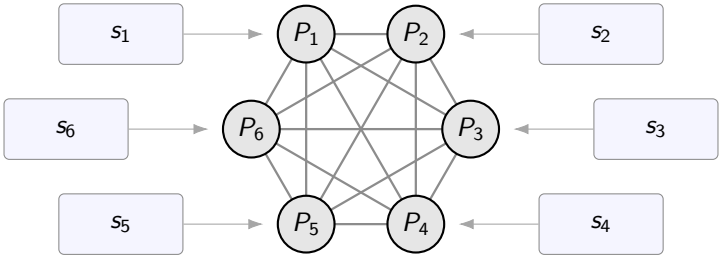
Un protocollo di *secret sharing* è una coppia di algoritmi $\Pi = (\mathrm{SS}, \mathrm{Rec})$ attraverso cui un *dealer* distribuisce un segreto m tra n giocatori in modo che:

- piccole coalizioni non apprendano nulla sul segreto m;
- insiemi abbastanza grandi di giocatori possano ricostruire completamente il segreto;



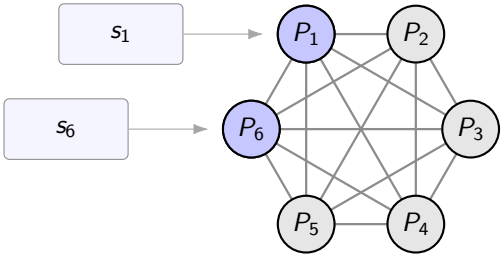
t-Secret Sharing: obiettivo e definizione

- **Distribuzione delle *share*:** il *dealer* calcola $(s_1, s_2, \ldots, s_n) \leftarrow \mathrm{SS}(1^\lambda, m)$ e consegna a ogni giocatore P_i la propria *share*.



t-Secret Sharing: obiettivo e definizione

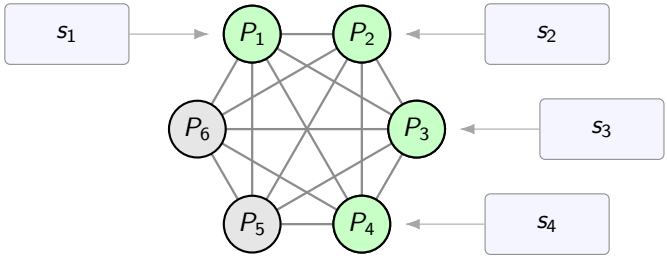
- **Privacy:** per ogni coalizione $B \subseteq [n]$ con $|B| \leq t$, le *share* $\{s_i\}_{i \in B}$ non rivelano alcuna informazione sul segreto. Comunque vengano scelti due segreti m, m', le rispettive *share* $(s_i)_{i \in B} \leftarrow \mathrm{SS}(1^\lambda, m)$ e $(s'_i)_{i \in B} \leftarrow \mathrm{SS}(1^\lambda, m')$ sono **statisticamente indistinguibili**.



t-Secret Sharing: obiettivo e definizione

- **Ricostruzione:** ogni coalizione sufficientemente grande di giocatori può ricostruire completamente il segreto. Formalmente, se $B \subseteq [n]$ e $|B| \geq t+1$, allora

$$\mathrm{Rec}(\{s_i\}_{i \in B}) = m.$$

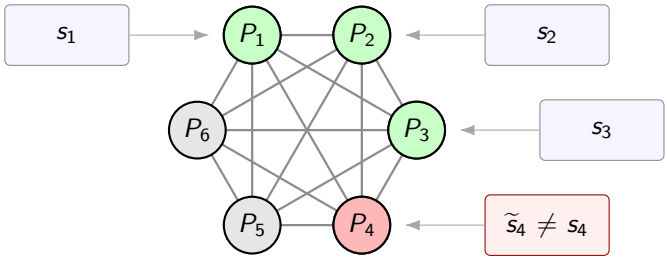


E se ci fossero degli impostori?

- In un protocollo di *secret sharing*, la ricostruzione è garantita solo se i giocatori coinvolti trasmettono onestamente le proprie *share*, senza modificarle.

E se ci fossero degli impostori?

- In un protocollo di *secret sharing*, la ricostruzione è garantita solo se i giocatori coinvolti trasmettono onestamente le proprie *share*, senza modificarle.
- Ma se uno di loro è corrotto, può sostituire la propria *share* con un valore arbitrario, ad esempio $\widetilde{s}_4 \neq s_4$. Conseguentemente, si potrebbe avere $\mathrm{Rec}(s_1, s_2, s_3, \tilde{s}_4) \neq m$.



δ-Robust Secret Sharing: obiettivo e definizione

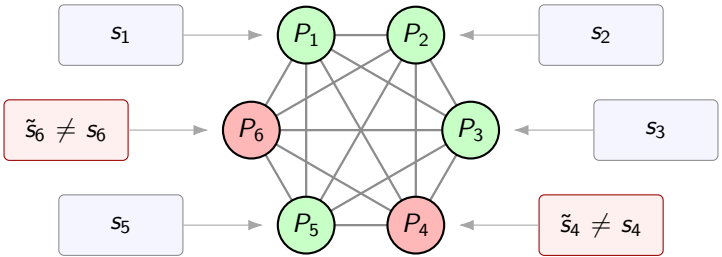
Vorremmo che, anche se un insieme A di giocatori corrotti modifichi le proprie *share*, la ricostruzione che coinvolge tutte le *share* resti corretta.

δ-Robust Secret Sharing: obiettivo e definizione

Vorremmo che, anche se un insieme A di giocatori corrotti modifichi le proprie *share*, la ricostruzione che coinvolge tutte le *share* resti corretta.

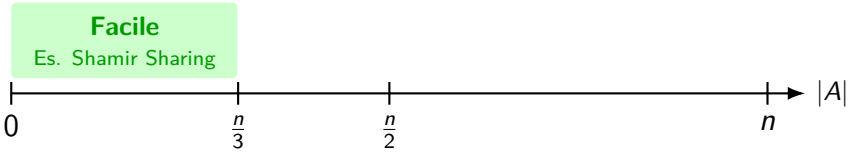
- δ-**robustezza:** dette $(s_i)_{i\in[n]} \leftarrow \mathrm{SS}(1^\lambda, m)$ le *share* corrette e $(\tilde{s}_i)_{i\in[n]}$ quelle dopo l'eventuale manomissione, con $\tilde{s}_i = s_i$ per ogni $i \notin A$, si abbia che:

$$\Pr\left[\mathrm{Rec}(\{\tilde{s}_i\}_{i\in[n]}) \neq m\right] < \delta$$



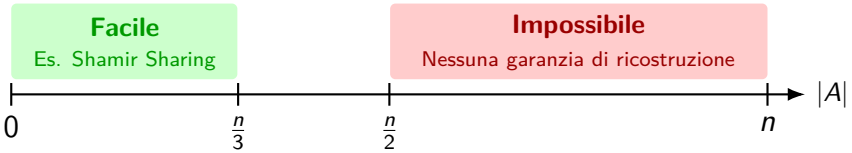
Quando la robustezza è possibile e come ottenerla

Lo stato dell'arte varia in base al numero di impostori tollerabili $|A|$ rispetto al totale dei giocatori n coinvolti nel protocollo:



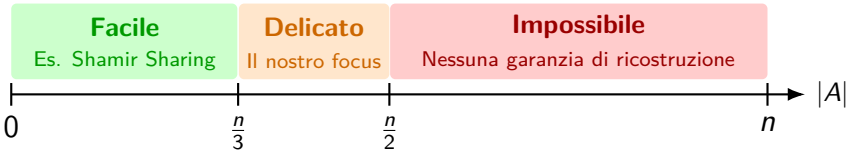
Quando la robustezza è possibile e come ottenerla

Lo stato dell'arte varia in base al numero di impostori tollerabili $|A|$ rispetto al totale dei giocatori n coinvolti nel protocollo:



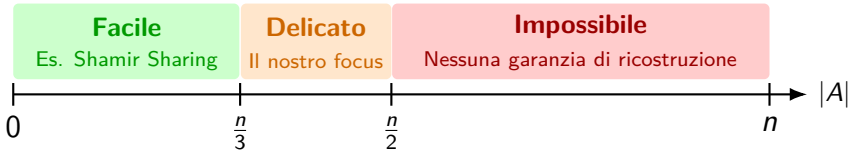
Quando la robustezza è possibile e come ottenerla

Lo stato dell'arte varia in base al numero di impostori tollerabili $|A|$ rispetto al totale dei giocatori n coinvolti nel protocollo:



Quando la robustezza è possibile e come ottenerla

Lo stato dell'arte varia in base al numero di impostori tollerabili $|A|$ rispetto al totale dei giocatori n coinvolti nel protocollo:



Il Regime Intermedio ($n/3 < |A| < n/2$)

In questa fascia, i protocolli di **Robust Secret Sharing** si ottengono aggiungendo **meccanismi di autenticazione** a uno schema standard.

Obiettivo: Rilevare o neutralizzare le *share* manomesse.

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

Esempio di one-time MAC su $\mathbb{F}_q$

Si sceglie una chiave casuale $k = (a, b) \leftarrow \mathbb{F}_q^2$ e si definisce $\mathrm{MAC}(k, m) = am + b$. Osservare una sola coppia $(m, \tau = \mathrm{MAC}(k, m))$ non permette, senza conoscere k, di produrre un tag valido $\tilde{\tau}$ per un messaggio diverso $\tilde{m} \neq m$, se non con probabilità $1/q$.

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$

dealer $\to P_j$ **:**

- s_j
- $\tau_{i,j}$

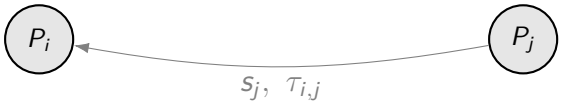
MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$



dealer $\to P_j$ **:**

- s_j
- $\tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

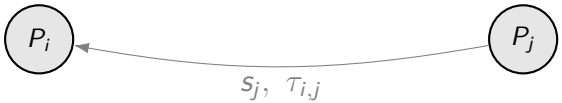
- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$

verifica di P_i su P_j:
$\tau_{i,j} \stackrel{?}{=} \mathrm{MAC}(k_{i,j}, s_j)$



dealer $\to P_j$ **:**

- s_j
- $\tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

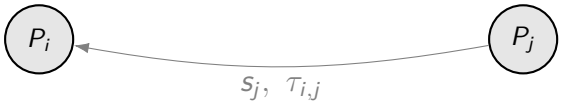
- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

verifica di P_i su P_j:
$\tau_{i,j} \stackrel{?}{=} \mathrm{MAC}(k_{i,j}, s_j)$



dealer $\to P_j$ **:**

- s_j
- $k_{j,i}$
- $\tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

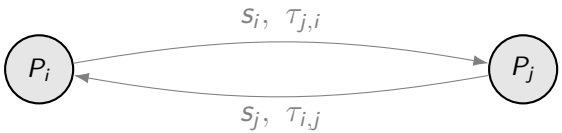
- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

verifica di P_i su P_j:
$\tau_{i,j} \stackrel{?}{=} \mathrm{MAC}(k_{i,j},\, s_j)$



dealer $\to P_j$ **:**

- s_j
- $k_{j,i}$
- $\tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

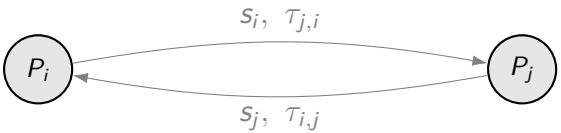
- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

verifica di P_i su P_j:
$\tau_{i,j} \stackrel{?}{=} \mathrm{MAC}(k_{i,j},\, s_j)$



dealer $\to P_j$ **:**

- s_j
- $k_{j,i}$
- $\tau_{i,j}$

verifica di P_j su P_i:
$\tau_{j,i} \stackrel{?}{=} \mathrm{MAC}(k_{j,i},\, s_i)$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j}, s_j).$$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

Impostore

dealer $\to P_j$ **:**

- $\tilde{s}_j \neq s_j$
- $k_{j,i}$
- $\tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

Impostore

dealer $\to P_j$ **:**

- $\tilde{s}_j \neq s_j$
- $k_{j,i}$
- $\tilde{\tau}_{i,j} \neq \tau_{i,j}$

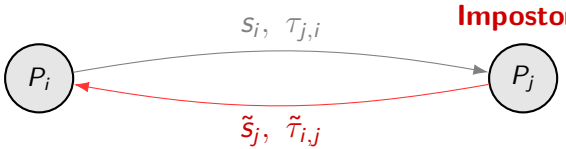
MAC locali: idea su una coppia (P_i, P_j)

- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$



dealer $\to P_j$ **:**

- $\tilde{s}_j \neq s_j$
- $k_{j,i}$
- $\tilde{\tau}_{i,j} \neq \tau_{i,j}$

MAC locali: idea su una coppia (P_i, P_j)

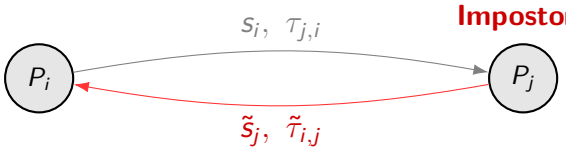
- **Idea:** il dealer distribuisce le *share* prodotte, inoltre consegna ad ogni giocatore sia delle *chiavi* per controllare le *share* degli altri, sia dei *tag* per essere controllato dagli altri.
- La ricostruzione viene eseguita da ogni singolo giocatore P_i, che utilizza solo le *share* che ritiene inalterate, ossia tutte e sole le s_j per cui

$$\tau_{i,j} = \mathrm{MAC}(k_{i,j},\, s_j).$$

dealer $\to P_i$ **:**

- s_i
- $k_{i,j}$
- $\tau_{j,i}$

verifica di P_i su P_j:
$\tilde{\tau}_{i,j} \neq \mathrm{MAC}(k_{i,j},\, \tilde{s}_j)$

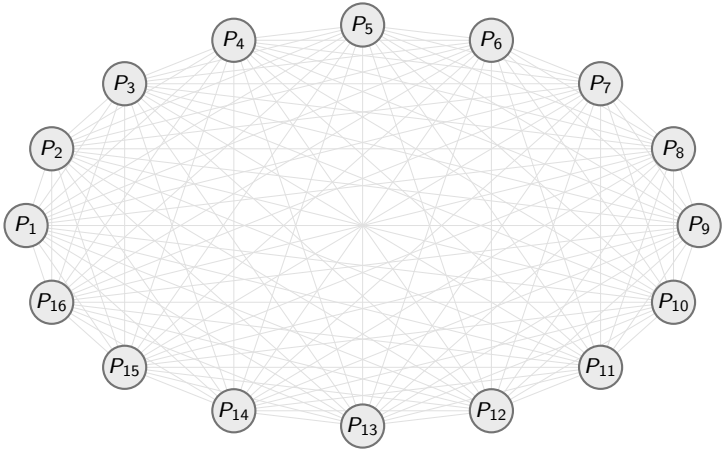


dealer $\to P_j$ **:**

- $\tilde{s}_j \neq s_j$
- $k_{j,i}$
- $\tilde{\tau}_{i,j} \neq \tau_{i,j}$

verifica di P_j su P_i:
$\tau_{j,i} \stackrel{?}{=} \mathrm{MAC}(k_{j,i},\, s_i)$

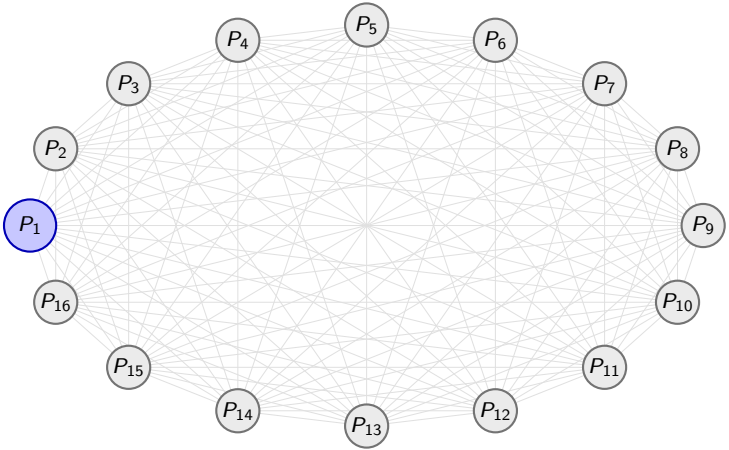
Se ogni giocatore controlla tutti gli altri...



Se ogni giocatore controlla tutti gli altri...

dealer $\rightarrow P_1$

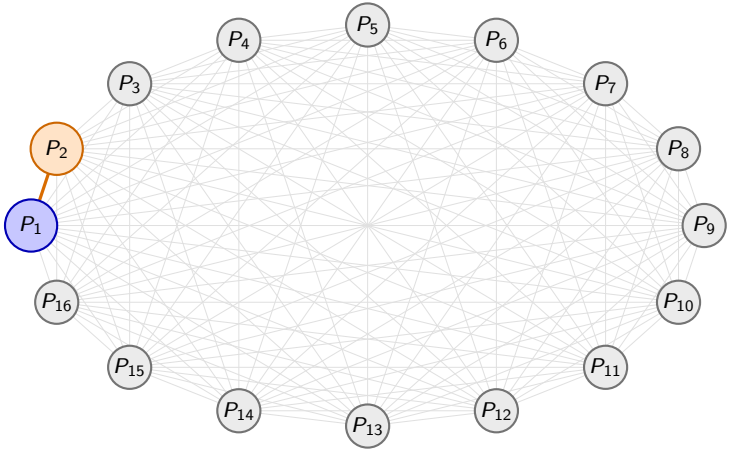
- s_1



Se ogni giocatore controlla tutti gli altri...

dealer $\rightarrow P_1$

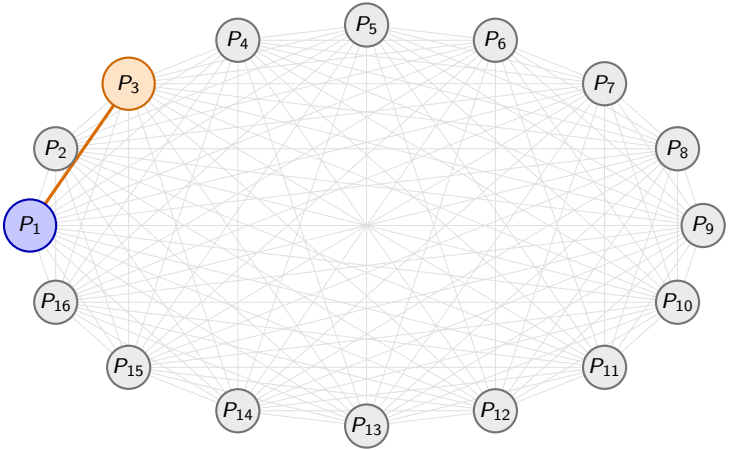
- s_1
- $k_{1,2}$
- $\tau_{2,1}$



Se ogni giocatore controlla tutti gli altri...

dealer $\rightarrow P_1$

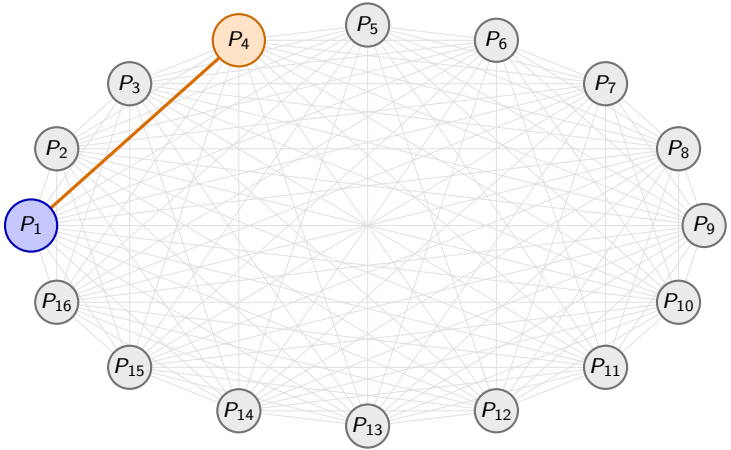
- s_1
- $k_{1,2}$
- $\tau_{2,1}$
- $k_{1,3}$
- $\tau_{3,1}$



Se ogni giocatore controlla tutti gli altri...

dealer $\rightarrow P_1$

- s_1
- $k_{1,2}$
- $\tau_{2,1}$
- $k_{1,3}$
- $\tau_{3,1}$
- $k_{1,4}$
- $\tau_{4,1}$

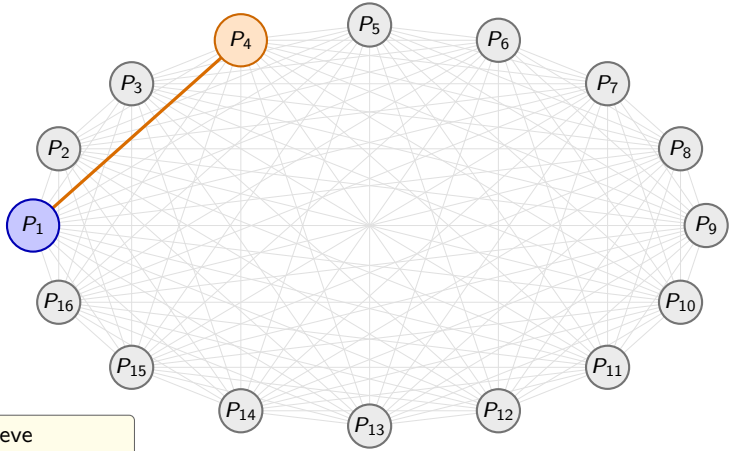


Se ogni giocatore controlla tutti gli altri...

dealer $\rightarrow P_1$

- s_1
- $k_{1,2}$
- $\tau_{2,1}$
- $k_{1,3}$
- $\tau_{3,1}$
- $k_{1,4}$
- $\tau_{4,1}$

$\vdots$

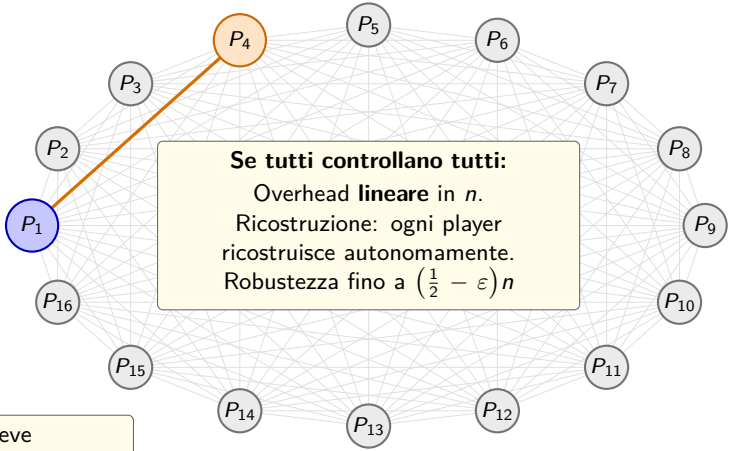


In generale, P_1 riceve
$s_1,\ \{k_{1,j}\}_{j\neq 1},\ \{\tau_{j,1}\}_{j\neq 1}$.

Se ogni giocatore controlla tutti gli altri...

dealer $\to P_1$

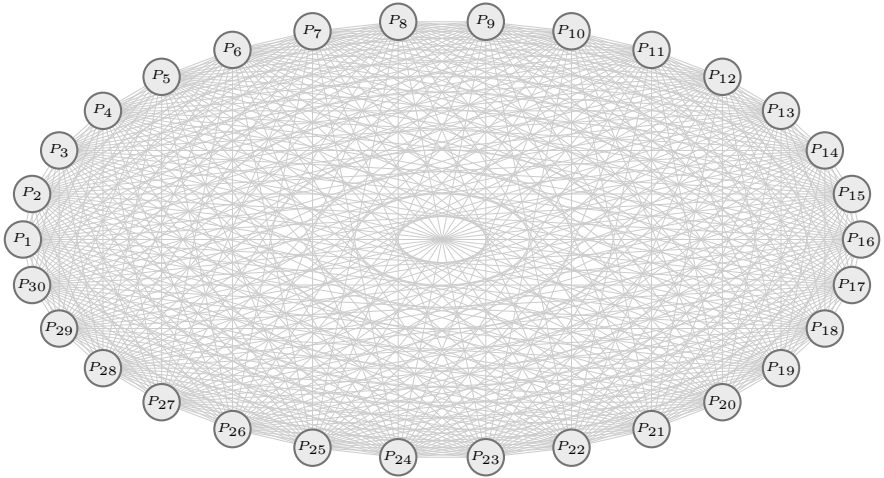
- s_1
- $k_{1,2}$
- $\tau_{2,1}$
- $k_{1,3}$
- $\tau_{3,1}$
- $k_{1,4}$
- $\tau_{4,1}$
- $\vdots$



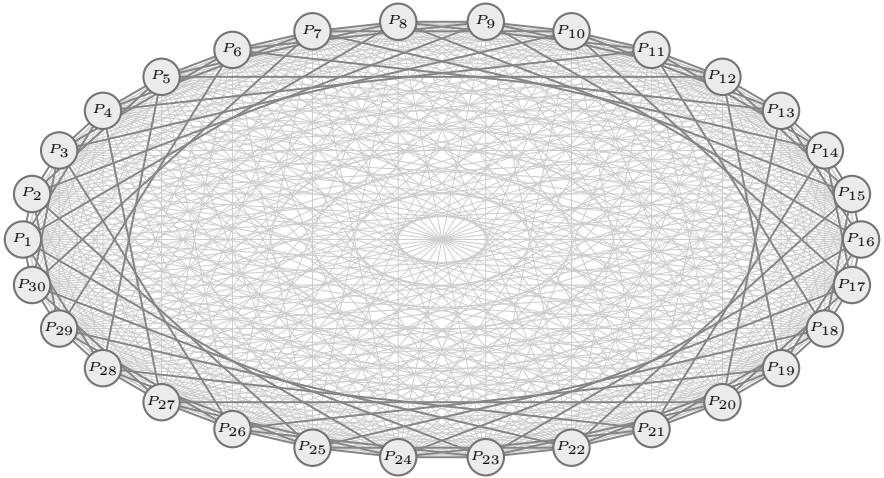
Se tutti controllano tutti:
Overhead **lineare** in n.
Ricostruzione: ogni player ricostruisce autonomamente.
Robustezza fino a $\left(\frac{1}{2} - \varepsilon\right)n$

In generale, P_1 riceve
$s_1,\ \{k_{1,j}\}_{j\neq 1},\ \{\tau_{j,1}\}_{j\neq 1}$.

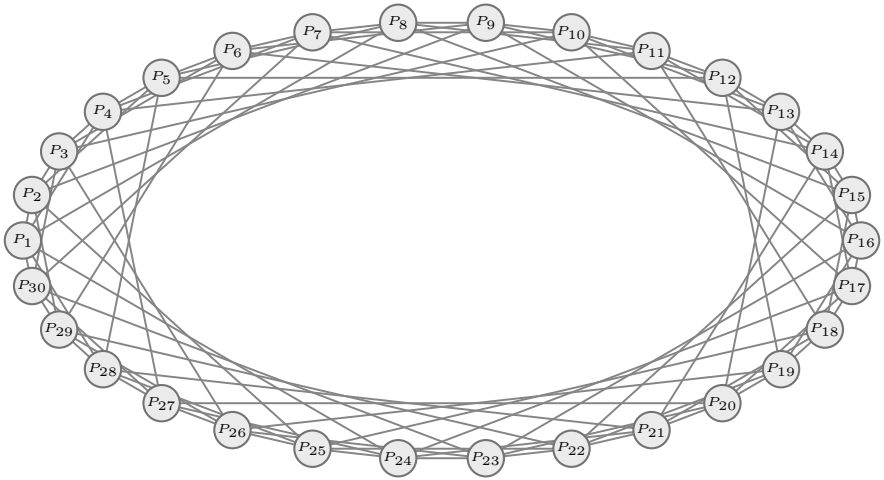
E se ogni giocatore non controllasse tutti gli altri, ma solo pochi vicini in un grafo sparso?



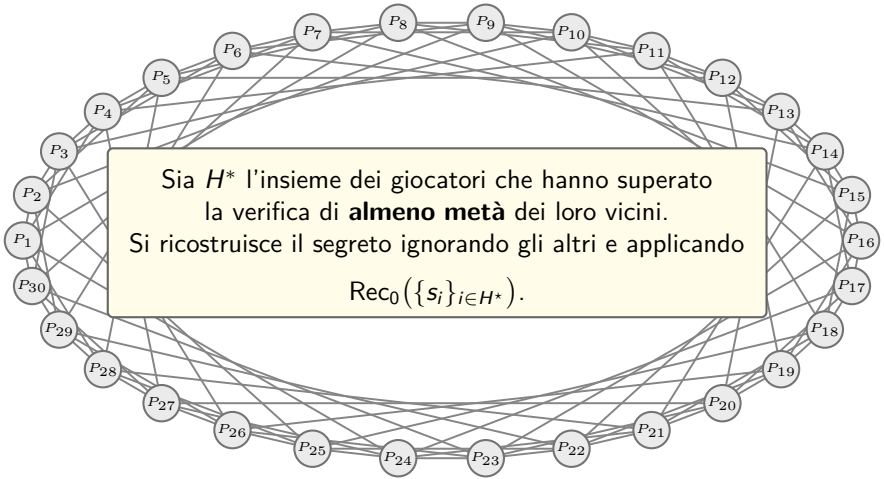
E se ogni giocatore non controllasse tutti gli altri, ma solo pochi vicini in un grafo sparso?



E se ogni giocatore non controllasse tutti gli altri, ma solo pochi vicini in un grafo sparso?



E se ogni giocatore non controllasse tutti gli altri, ma solo pochi vicini in un grafo sparso?



Sia H^* l'insieme dei giocatori che hanno superato la verifica di **almeno metà** dei loro vicini.
Si ricostruisce il segreto ignorando gli altri e applicando

$$\mathrm{Rec}_0\left(\{s_i\}_{i\in H^*}\right).$$

$(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è robusto

Teorema 3 (Hemenway–Ostrovsky, *"Efficient RSS from Expander Graphs"*, 2016)

Sia G un grafo d-regolare su n vertici. Siano (SS_0, Rec_0) uno schema di *Secret Sharing* e MAC un metodo di autenticazione sufficientemente sicuri. Per ogni $\epsilon > 0$

$(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è robusto

Teorema 3 (Hemenway–Ostrovsky, *"Efficient RSS from Expander Graphs"*, 2016)

Sia G un grafo d-regolare su n vertici. Siano (SS_0, Rec_0) uno schema di *Secret Sharing* e MAC un metodo di autenticazione sufficientemente sicuri. Per ogni $\epsilon > 0$

Condizione Spettrale: se il grafo è un buon expander, ovvero se $\sigma_2(A)$ soddisfa:

$$\sigma_2(A) \leq \sqrt{d \cdot 4\varepsilon^4} \cdot \sqrt{d}$$

$(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è robusto

Teorema 3 (Hemenway–Ostrovsky, *"Efficient RSS from Expander Graphs"*, 2016)

Sia G un grafo d-regolare su n vertici. Siano (SS_0, Rec_0) uno schema di *Secret Sharing* e MAC un metodo di autenticazione sufficientemente sicuri. Per ogni $\epsilon > 0$

Condizione Spettrale: se il grafo è un buon expander, ovvero se $\sigma_2(A)$ soddisfa:

$$\sigma_2(A) \leq \sqrt{d \cdot 4\varepsilon^4} \cdot \sqrt{d}$$

Garanzie del Protocollo: allora la costruzione $(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è un protocollo con:

$(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è robusto

Teorema 3 (Hemenway–Ostrovsky, *"Efficient RSS from Expander Graphs"*, 2016)

Sia G un grafo d-regolare su n vertici. Siano (SS_0, Rec_0) uno schema di *Secret Sharing* e MAC un metodo di autenticazione sufficientemente sicuri. Per ogni $\epsilon > 0$

Condizione Spettrale: se il grafo è un buon expander, ovvero se $\sigma_2(A)$ soddisfa:

$$\sigma_2(A) \leq \sqrt{d \cdot 4\varepsilon^4} \cdot \sqrt{d}$$

Garanzie del Protocollo: allora la costruzione $(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è un protocollo con:

- **Alta Robustezza:** tollera fino a $\left(\frac{1}{2} - \varepsilon\right) n$ impostori;

$(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è robusto

Teorema 3 (Hemenway–Ostrovsky, *"Efficient RSS from Expander Graphs"*, 2016)

Sia G un grafo d-regolare su n vertici. Siano (SS_0, Rec_0) uno schema di *Secret Sharing* e MAC un metodo di autenticazione sufficientemente sicuri. Per ogni $\epsilon > 0$

Condizione Spettrale: se il grafo è un buon expander, ovvero se $\sigma_2(A)$ soddisfa:

$$\sigma_2(A) \leq \sqrt{d \cdot 4\varepsilon^4} \cdot \sqrt{d}$$

Garanzie del Protocollo: allora la costruzione $(SS^{G,\mathrm{MAC}}, \mathrm{Rec}^{G,\mathrm{MAC}})$ è un protocollo con:

- **Alta Robustezza:** tollera fino a $\left(\frac{1}{2} - \varepsilon\right) n$ impostori;
- **Efficienza:** la dimensione delle *share* è contenuta, pari a $s_0 + d(\log|\mathrm{Tag}| + \log|\mathrm{Key}|)$

Idea della dimostrazione: il mixing contro A

Il setup: Sia A l'insieme dei corrotti ($|A| = t$) e $H^\star$ i giocatori ritenuti onesti.
Sia $M = M_1 \cup M_2$ l'insieme dei giocatori **classificati erroneamente**:

Idea della dimostrazione: il mixing contro A

Il setup: Sia A l'insieme dei corrotti ($|A| = t$) e $H^\star$ i giocatori ritenuti onesti.
Sia $M = M_1 \cup M_2$ l'insieme dei giocatori **classificati erroneamente**:

- M_1: giocatori onesti scartati per errore;

Idea della dimostrazione: il mixing contro A

Il setup: Sia A l'insieme dei corrotti ($|A| = t$) e $H^\star$ i giocatori ritenuti onesti.
Sia $M = M_1 \cup M_2$ l'insieme dei giocatori **classificati erroneamente**:

- M_1: giocatori onesti scartati per errore;
- M_2: impostori accettati per errore.

Idea della dimostrazione: il mixing contro A

Il setup: Sia A l'insieme dei corrotti ($|A| = t$) e $H^\star$ i giocatori ritenuti onesti.
Sia $M = M_1 \cup M_2$ l'insieme dei giocatori **classificati erroneamente**:

- M_1: giocatori onesti scartati per errore;
- M_2: impostori accettati per errore.

Obiettivo: Lo schema di base $(\mathsf{SS}_0, \mathsf{Rec}_0)$ ricostruisce se M_2 è sufficientemente piccolo.

Idea della dimostrazione: il mixing contro A

Il setup: Sia A l'insieme dei corrotti ($|A| = t$) e $H^\star$ i giocatori ritenuti onesti.
Sia $M = M_1 \cup M_2$ l'insieme dei giocatori **classificati erroneamente**:

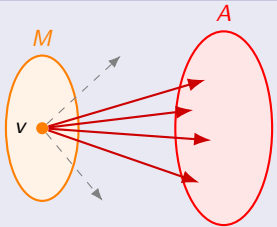
- M_1: giocatori onesti scartati per errore;
- M_2: impostori accettati per errore.

Obiettivo: Lo schema di base $(\mathrm{SS}_0, \mathrm{Rec}_0)$ ricostruisce se M_2 è sufficientemente piccolo.

Dove entra l'Expander Mixing Lemma

Se un nodo è in M, significa che **almeno metà dei suoi vicini** ha falsificato la verifica, e quindi questi sono in A.
Ne segue che

$$\frac{d}{2}|M| \leq |E(M,A)| \overset{\text{E.M.L.}}{\leq} \frac{d}{n}|M||A| + \sigma_2\sqrt{|M||A|}$$



Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).

Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).

- **Il Potere del Mixing:** grazie all'**Expander Mixing Lemma**, i grafi expander garantiscono una distribuzione degli archi uniforme e pseudo-casuale.

Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).
- **Il Potere del Mixing:** grazie all'**Expander Mixing Lemma**, i grafi expander garantiscono una distribuzione degli archi uniforme e pseudo-casuale.
- **Il Salto in Crittografia:** questa "casualità strutturata" diventa lo scudo matematico perfetto contro avversari potenti, abilitando la costruzione di:

Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).
- **Il Potere del Mixing:** grazie all'**Expander Mixing Lemma**, i grafi expander garantiscono una distribuzione degli archi uniforme e pseudo-casuale.
- **Il Salto in Crittografia:** questa "casualità strutturata" diventa lo scudo matematico perfetto contro avversari potenti, abilitando la costruzione di:
  - protocolli di **Robust Secret Sharing**;

Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).
- **Il Potere del Mixing:** grazie all'**Expander Mixing Lemma**, i grafi expander garantiscono una distribuzione degli archi uniforme e pseudo-casuale.
- **Il Salto in Crittografia:** questa "casualità strutturata" diventa lo scudo matematico perfetto contro avversari potenti, abilitando la costruzione di:
  - protocolli di **Robust Secret Sharing**;
  - **Non-Malleable Codes** nel modello *Split-State*.

Cosa portarsi a casa

- **La Teoria Spettrale:** gli autovalori non sono solo quantità algebriche, ma catturano in modo profondo la **geometria globale** del grafo (connettività, espansione, tagli).
- **Il Potere del Mixing:** grazie all'**Expander Mixing Lemma**, i grafi expander garantiscono una distribuzione degli archi uniforme e pseudo-casuale.
- **Il Salto in Crittografia:** questa "casualità strutturata" diventa lo scudo matematico perfetto contro avversari potenti, abilitando la costruzione di:
  - protocolli di **Robust Secret Sharing**;
  - **Non-Malleable Codes** nel modello *Split-State*.

Grazie per l'attenzione!

Essere sparsi è bene, ma expandersi è meglio

1973/1988 — Costruzioni esplicite e Ramanujan graphs

Margulis / Sarnak

Nascono le prime famiglie **esplicite** di expander: l'espansione smette di essere solo un fenomeno probabilistico e diventa qualcosa che si può costruire in modo concreto.

Gregory Margulis

Peter Sarnak

1973/88
Origini

Essere sparsi è bene, ma expandersi è meglio

1985 — Il ponte spettrale

Noga Alon & Vitali D. Milman

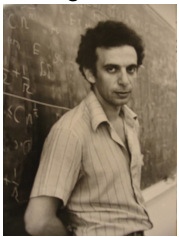
La spectral graph theory rende quantitativo il legame tra **autovalori** e **tagli del grafo**.

Lo spettro della matrice di a misura anche robustezza combinatoria, connettività e qualità dell'espansione.

1973/88
Origini

1985
Spettro & tagli

Noga Alon



Vitali D. Milman

Essere sparsi è bene, ma expandersi è meglio

2002 — Zig-zag product

Omer Reingold, Salil Vadhan, Avi Wigderson **Gödel 2009**

Lo **zig-zag product** mostra come costruire e comporre expander.

Queste idee confluiscono poi anche nel risultato **undirected connectivity in log-space**.

Gli expander diventano uno strumento fondamentale in derandomizzazione e complessità computazionale.

Avi Wigderson

Salil Vadhan

Omer Reingold

Essere sparsi è bene, ma expandersi è meglio

2006 — Una nuova prova del PCP Theorem

Irit Dinur **Gödel 2019**

Il **PCP Theorem** era già stato dimostrato negli anni '90.

Dinur ne dà una prova più combinatoria e accessibile, basata su **gap amplification**, dove gli expander entrano in modo strutturale.

Gli expander sono al cuore della *hardness of approximation*.

Irit Dinur

Essere sparsi è bene, ma expandersi è meglio

2011 — Dallo spettro agli algoritmi veloci

Daniel A. Spielman & Shang-Hua Teng **Gödel 2015**

Spectral sparsification: è possibile sostituire un grafo con uno molto più sparso preservandone quasi la stessa geometria spettrale.
Le idee spettrali nate nello studio degli expander diventano strumenti algoritmici di enorme impatto.

Daniel A. Spielman & Shang-Hua Teng

Un altro problema di cui gli autovalori sono soluzione

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

Un altro problema di cui gli autovalori sono soluzione

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(2) Grafi bipartiti e MaxCut

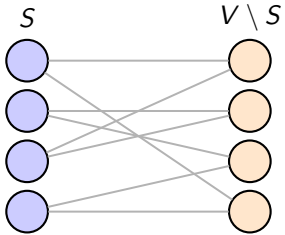
G bipartito $\iff \lambda_n(A) = -d$ $\qquad$ $\textsc{MaxCut}(G) = \max_{S \subset V} |E(S, V \setminus S)| \leq \frac{n}{4}(d - \lambda_n(A))$

Un altro problema di cui gli autovalori sono soluzione

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(2) Grafi bipartiti e MaxCut

G bipartito $\iff \lambda_n(A) = -d$ $\qquad$ $\textsc{MaxCut}(G) = \max_{S \subset V} |E(S, V \setminus S)| \leq \frac{n}{4}(d - \lambda_n(A))$

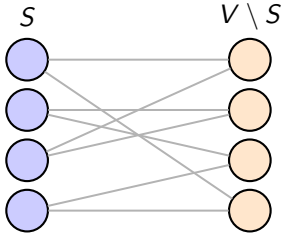


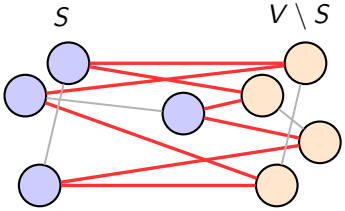
Un altro problema di cui gli autovalori sono soluzione

Spettro di A : $\quad d = \lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) \geq -d.$

(2) Grafi bipartiti e MaxCut

G bipartito $\iff \lambda_n(A) = -d$ $\qquad$ $\textsc{MaxCut}(G) = \max_{S \subset V} |E(S, V \setminus S)| \leq \frac{n}{4}(d - \lambda_n(A))$





Autovettori e autovalori: background

Sia $M \in \mathbb{R}^{n \times n}$. Un *autovettore* $x \neq 0$ con *autovalore* $\lambda \in \mathbb{R}$ soddisfa:

$$Mx = \lambda x.$$

Teorema Spettrale

Sia $M \in \mathbb{R}^{n \times n}$ una matrice simmetrica, allora

- $\exists D \in \mathbb{R}^{n \times n}$ invertibile tale che $DMD^{-1} = \Lambda$ è una matrice diagonale.

Inoltre, se $DMD^{-1} = \Lambda$ è una matrice diagonale, allora

- le colonne di $D = (D_1|D_2|\ldots|D_n)$ formano una base di autovettori, ortogonali tra loro i cui rispettivi autovalori sono gli elementi sulla diagonale di Λ.

Quoziente di Rayleigh: lo spettro come problema di ottimizzazione

Per $M \in \mathbb{R}^{n\times n}$ simmetrica e $x \neq 0$, il *quoziente di Rayleigh* associato a M è:

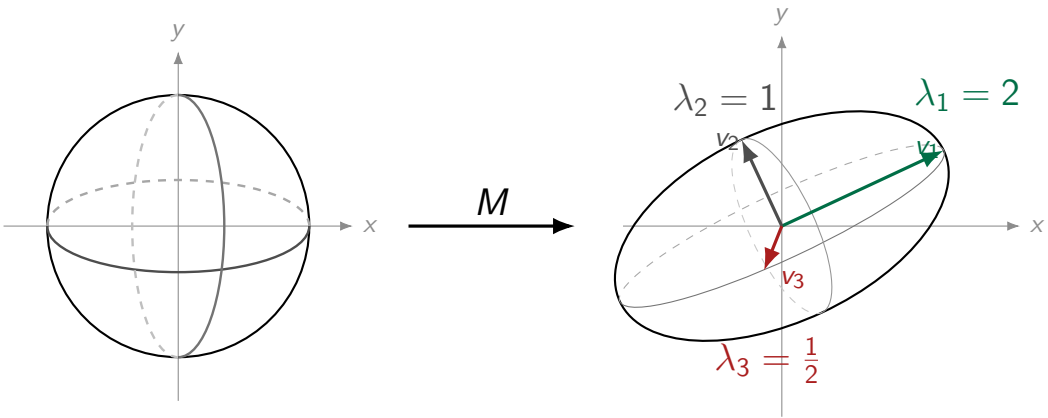
$$R_M(x) := \frac{x^\top M x}{x^\top x}.$$

Principio variazionale

Sia $M \in \mathbb{R}^{n\times n}$ una matrice simmetrica con spettro $\lambda_1 \geq \cdots \geq \lambda_n$ e autovettori associati $x_1, \ldots, x_n$, allora

$$\lambda_1 = R_M(x_1) = \max_{x\neq 0} R_M(x), \qquad \lambda_2 = R_M(x_2) = \max_{\substack{x\neq 0 \\ x\cdot x_1 = 0}} R_M(x), \qquad \ldots \qquad \lambda_n = \min_{x\neq 0} R_M(x).$$

Significato geometrico del quoziente di Rayleigh



Expander Mixing Lemma: idea della dimostrazione

Claim: $\forall\, S, T \subseteq V \qquad \left| E(S,T) - \frac{d|S||T|}{n} \right| \;\leq\; \sigma \sqrt{|S||T|}$

- **Cambiare matrice:** sia $\sigma_2(A) = \max\{|\lambda_2(A)|, |\lambda_n(A)|\}$, sia $B \;:=\; A - \frac{d}{n}\mathbf{1}_{n\times n}$, allora

$$\sigma_2(A) \;=\; \max_{\sum x_v = 0} \frac{|x^\top A x|}{\|x\|^2} \;=\; \max_{x \neq 0} \frac{|x^\top B x|}{\|x\|^2} \;=\; \max_{x,y \neq 0} \frac{|x^\top B y|}{\|x\|\,\|y\|}.$$

- **Vettori indicatori:** siano $S, T \subset V$ e $\mathbb{1}_S, \mathbb{1}_T \in \{0,1\}^n$,

$$\mathbb{1}_S^\top B\, \mathbb{1}_T \;=\; \mathbb{1}_S^\top (A - \frac{d}{n}\mathbf{1}_{n\times n})\, \mathbb{1}_T = \; E(S,T) - \frac{d|S||T|}{n}.$$

Quindi,

$$\frac{\left|\mathbb{1}_S^\top B\, \mathbb{1}_T\right|}{\|\mathbb{1}_S\| \cdot \|\mathbb{1}_T\|} \leq \max_{x,y \in \{0,1\}^n} \frac{|x^\top B y|}{\|x\|\,\|y\|} \leq \max_{x,y \neq 0} \frac{|x^\top B y|}{\|x\|\,\|y\|} = \sigma_2(A)$$

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Distribuzione: l'algoritmo $\mathrm{SS}^{G,\mathrm{MAC}}(1^\lambda, m)$

Il dealer:

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Distribuzione: l'algoritmo $\mathrm{SS}^{G,\mathrm{MAC}}(1^\lambda, m)$

Il dealer:

- calcola $(s_1, \ldots, s_n) \leftarrow SS_0(1^\lambda, m)$;

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Distribuzione: l'algoritmo $\mathrm{SS}^{G,\mathrm{MAC}}(1^\lambda, m)$

Il dealer:

- calcola $(s_1, \ldots, s_n) \leftarrow SS_0(1^\lambda, m)$;
- per ogni $i \in [n]$ e $j \in \Gamma(i)$, estrae una chiave k_{ij} e calcola $\tau_{ij} = \mathrm{MAC}(k_{ij}, s_j)$;

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Distribuzione: l'algoritmo $\mathrm{SS}^{G,\mathrm{MAC}}(1^\lambda, m)$

Il dealer:

- calcola $(s_1, \ldots, s_n) \leftarrow SS_0(1^\lambda, m)$;
- per ogni $i \in [n]$ e $j \in \Gamma(i)$, estrae una chiave k_{ij} e calcola $\tau_{ij} = \mathrm{MAC}(k_{ij}, s_j)$;
- consegna a P_i il pacchetto $\Big(s_i,\ \{k_{ij}\}_{j\in\Gamma(i)},\ \{\tau_{ji}\}_{j\in\Gamma(i)}\Big)$.

Protocollo RSS su un grafo d-regular

Sia G un grafo d-regolare sui giocatori $P_1, \ldots, P_n$, e sia $N(i)$ l'insieme dei vicini di P_i.

Distribuzione: l'algoritmo $\mathrm{SS}^{G,\mathrm{MAC}}(1^\lambda, m)$

Il dealer:

- calcola $(s_1, \ldots, s_n) \leftarrow SS_0(1^\lambda, m)$;
- per ogni $i \in [n]$ e $j \in \Gamma(i)$, estrae una chiave k_{ij} e calcola $\tau_{ij} = \mathrm{MAC}(k_{ij}, s_j)$;
- consegna a P_i il pacchetto $\Big(s_i,\ \{k_{ij}\}_{j\in\Gamma(i)},\ \{\tau_{ji}\}_{j\in\Gamma(i)}\Big)$.

Ricostruzione: l'algoritmo $\mathrm{Rec}^{G,\mathrm{MAC}}$

Si definisce H^* l'insieme dei giocatori che hanno superato la verifica di almeno metà dei loro vicini. Si ricostruisce il segreto ignorando gli altri e applicando $\mathrm{Rec}_0(\{s_i\}_{i\in H^*})$.

Dallo spettro alla crittografia

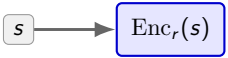
Non-malleable codes

Il mixing quasi casuale degli expander diventa robustezza contro avversari.

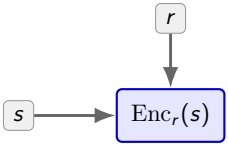
Codici classici

s

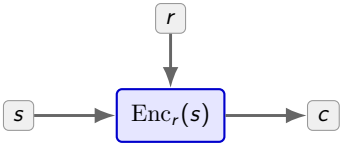
Codici classici



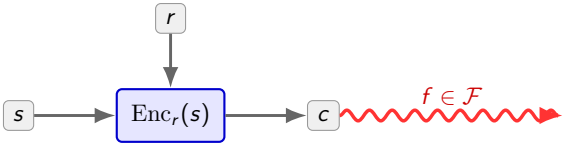
Codici classici



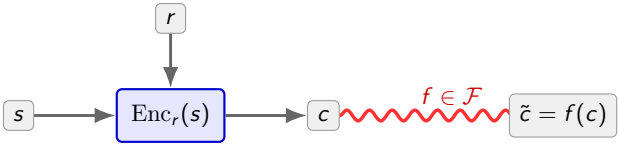
Codici classici



Codici classici

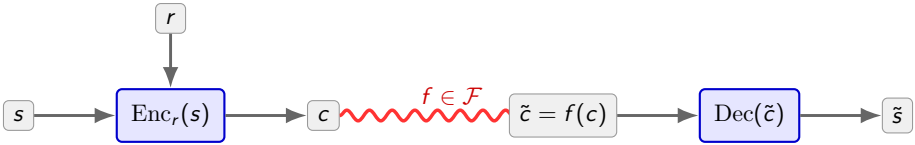


Codici classici

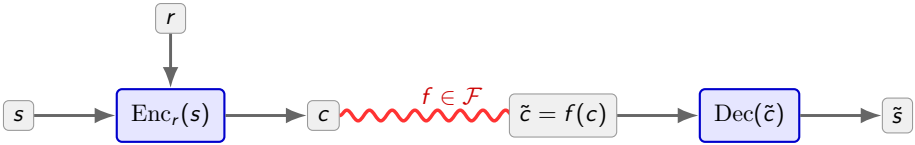


Codici classici

Codici classici

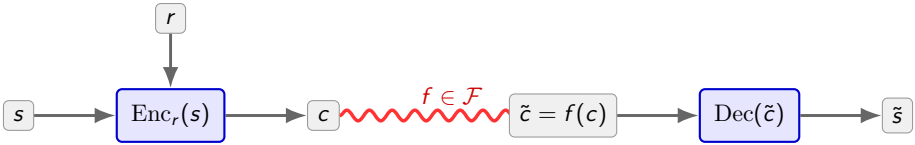


Codici classici



Garanzie di $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}$:

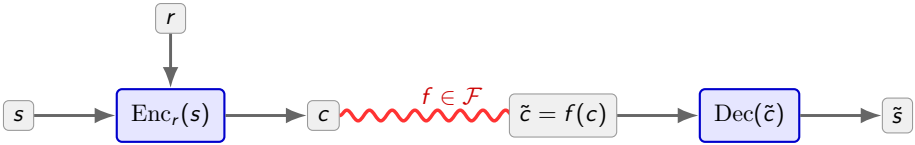
Codici classici



Garanzie di $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}$:

- **Error-correcting:**
  $\forall s, \forall f \in \mathcal{F}$ si ha che $\tilde{s} = s$

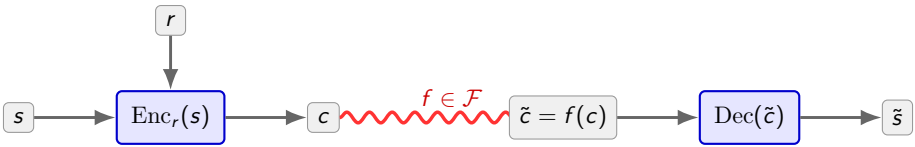
Codici classici



Garanzie di $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}$:

- **Error-correcting:**
  $\forall s, \forall f \in \mathcal{F}$ si ha che $\tilde{s} = s$
- **Error-detecting:**
  $\forall s, \forall f \in \mathcal{F}$ si ha che $\tilde{s} = s$ oppure $\tilde{s} = \perp$

Codici classici e limiti contro il tampering



Garanzie di $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}$:

- **Error-correcting:**
  $\forall s, \forall f \in \mathcal{F}$ si ha che $\tilde{s} = s$
- **Error-detecting:**
  $\forall s, \forall f \in \mathcal{F}$ si ha che $\tilde{s} = s$ oppure $\tilde{s} = \perp$

Risultati di Impossibilità

Ottenere queste garanzie per famiglie $\mathcal{F}$ utili è matematicamente impossibile. Falliscono a causa di limiti come

- distanza di Hamming
- semplici funzioni costanti in $\mathcal{F}$, come $f \equiv c^* = \mathrm{Enc}(s^*, r^*)$

Il cambio di paradigma

Il problema: se l'avversario è troppo potente, ossia la famiglia $\mathcal{F}$ è sufficientemente grande, non possiamo impedire di modificare il *cyphertext* e ottenerne un altro valido.

Il cambio di paradigma: la Non-Malleabilità

Il problema: se l'avversario è troppo potente, ossia la famiglia $\mathcal{F}$ è sufficientemente grande, non possiamo impedire di modificare il *cyphertext* e ottenerne un altro valido.

Una soluzione (Dziembowski, Pietrzak, Wichs - 2010):
Rinunciamo a correggere o rilevare sempre l'errore. Chiediamo invece che, se il *cyphertext* viene alterato, il risultato $\tilde{s}$ sia **completamente indipendente** dall'originale.

Il cambio di paradigma: la Non-Malleabilità

Il problema: se l'avversario è troppo potente, ossia la famiglia $\mathcal{F}$ è sufficientemente grande, non possiamo impedire di modificare il *cyphertext* e ottenerne un altro valido.

Una soluzione (Dziembowski, Pietrzak, Wichs - 2010):
Rinunciamo a correggere o rilevare sempre l'errore. Chiediamo invece che, se il *cyphertext* viene alterato, il risultato $\tilde{s}$ sia **completamente indipendente** dall'originale.

Malleable code

L'avversario non conosce s, ma sa produrre un $\tilde{s}$ logicamente correlato ad esso.

Es: trasforma $s =$"Offri 100€" in $\tilde{s} =$"Offri 1000€" aggiungendo uno zero.

Il cambio di paradigma: la Non-Malleabilità

Il problema: se l'avversario è troppo potente, ossia la famiglia $\mathcal{F}$ è sufficientemente grande, non possiamo impedire di modificare il *cyphertext* e ottenerne un altro valido.

Una soluzione (Dziembowski, Pietrzak, Wichs - 2010):
Rinunciamo a correggere o rilevare sempre l'errore. Chiediamo invece che, se il *cyphertext* viene alterato, il risultato $\tilde{s}$ sia **completamente indipendente** dall'originale.

Malleable code

L'avversario non conosce s, ma sa produrre un $\tilde{s}$ logicamente correlato ad esso.

Es: trasforma $s =$"Offri 100€" in $\tilde{s} =$"Offri 1000€" aggiungendo uno zero.

Non-Malleable code

L'avversario può solo:

- lasciare il messaggio intatto ($\tilde{s} = s$);
- distruggerlo ($\tilde{s} = \perp$);
- sostituirlo con un $\tilde{s}$ **scorrelato** da s

Non-malleabilità: una definizione formale

Dopo il *tampering*, il messaggio decodificato $\tilde{s}$ è **o uguale** all'originale s, **o non dà nessuna informazione** riguardo ad esso.

Non-malleabilità: una definizione formale

Dopo il *tampering*, il messaggio decodificato $\tilde{s}$ è **o uguale** all'originale s, **o non dà nessuna informazione** riguardo ad esso.

Non-Malleable Codes rispetto a $\mathcal{F}$

Uno schema $(\mathrm{Enc}, \mathrm{Dec})$ è $\epsilon-$*non malleabile* rispetto a $\mathcal{F}$ se per ogni funzione $f \in \mathcal{F}$ esiste un **simulatore** D_f (efficientemente calcolabile), che ha output in $\{0,1\}^k \cup \{\perp, \mathrm{same}^\star\}$, tale che per ogni s:

$$\mathrm{Dec}(f(\mathrm{Enc}(s))) \approx_\epsilon \begin{cases} s & \text{se } \tilde{s} = \mathrm{same}^*,\ \tilde{s} \leftarrow D_f \\ \tilde{s} & \text{altrimenti, con } \tilde{s} \leftarrow D_f \end{cases}$$

Non-malleabilità: una definizione formale

Dopo il *tampering*, il messaggio decodificato $\tilde{s}$ è **o uguale** all'originale s, **o non dà nessuna informazione** riguardo ad esso.

Non-Malleable Codes rispetto a $\mathcal{F}$

Uno schema $(\mathrm{Enc}, \mathrm{Dec})$ è $\epsilon-$*non malleabile* rispetto a $\mathcal{F}$ se per ogni funzione $f \in \mathcal{F}$ esiste un **simulatore** D_f (efficientemente calcolabile), che ha output in $\{0,1\}^k \cup \{\perp, \mathsf{same}^\star\}$, tale che per ogni s:

$$\mathrm{Dec}(f(\mathrm{Enc}(s))) \approx_\epsilon \begin{cases} s & \text{se } \tilde{s} = \mathsf{same}^*,\ \tilde{s} \leftarrow D_f \\ \tilde{s} & \text{altrimenti, con } \tilde{s} \leftarrow D_f \end{cases}$$

Il ruolo cruciale del simulatore D_f:

- D_f genera la distribuzione dell'output dipendendo **solo da** f, senza conoscere l'originale s.

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

Memoria: s

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

execute(m_1)

Memoria: s

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

execute(m_1)

$G(s, m_1) = (s_1, a_1)$

Memoria: s

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

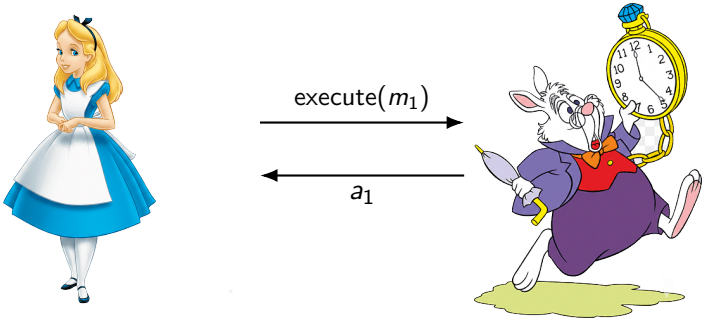
execute(m_1)

$G(s, m_1) = (s_1, a_1)$

Memoria: s_1

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.



$$G(s, m_1) = (s_1, a_1)$$

Memoria: s_1

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

Memoria: s_1

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

execute(m_2)

Memoria: s_1

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

execute(m_2)

$G(s_1, m_2) = (s_2, a_2)$

Memoria: s_1

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.

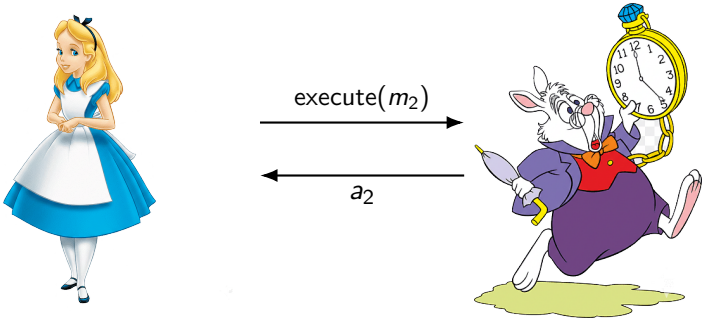
execute(m_2)

$G(s_1, m_2) = (s_2, a_2)$

Memoria: s_2

Honest interaction with a functionality

- **Interagire con una funzionalità:** data una funzionalità $G : \mathcal{S} \times \mathcal{M} \to \mathcal{S} \times \mathcal{A}$, un *giocatore onesto* interagisce inviando messaggi $m \in \mathcal{M}$. In base allo stato attuale $s \in \mathcal{S}$, viene calcolato il nuovo stato $s' \in \mathcal{S}$ e la risposta $a \in \mathcal{A}$.



$$G(s_1, m_2) = (s_2, a_2)$$

Memoria: s_2

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

Memoria: s

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

$f \in \mathcal{F}$

Memoria: s

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

$f \in \mathcal{F}$

Memoria: s

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

$f \in \mathcal{F}$

Memoria: s

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

$f \in \mathcal{F}$

Memoria: $\tilde{s} = f(s)$

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

Memoria: $\tilde{s}$

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

execute(m_1)

Memoria: $\tilde{s}$

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

execute(m_1)

$G(\tilde{s}, m_1) = (\tilde{s}_1, \tilde{a}_1)$

Memoria: $\tilde{S}$

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.

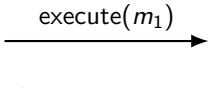
execute(m_1)

$G(\tilde{s}, m_1) = (\tilde{s}_1, \tilde{a}_1)$

Memoria: $\tilde{s}_1$

Tampering with the functionality

- **Attacchi attivi**: un *giocatore disonesto* vuole imparare lo stato iniziale, che è segreto. Per farlo, può *sabotare la funzionalità* alterando lo stato in memoria e interagendoci.



$G(\tilde{s}, m_1) = (\tilde{s}_1, \tilde{a_1})$

Memoria: $\tilde{s}_1$

A toy key-related attack via bit tampering

- **Esempio:** lo stato segreto è una chiave $k \in \{0,1\}^{\ell}$. La funzionalità risponde con $\mathrm{Enc}_k(m)$ senza cambiare la chiave k, inoltre vale $\mathrm{Enc}_k(m) = \perp \iff \mathrm{parity}(k) = 1$.

execute(m)

$\mathrm{Enc}_k(m) \neq \perp$

Memoria: $k_1\, k_2 \ldots k_{\ell}$

A toy key-related attack via bit tampering

- **Esempio:** lo stato segreto è una chiave $k \in \{0,1\}^{\ell}$. La funzionalità risponde con $\mathrm{Enc}_k(m)$ senza cambiare la chiave k, inoltre vale $\mathrm{Enc}_k(m) = \perp \iff \mathrm{parity}(k) = 1$.

$f_{k_1=0}$

Memoria: $k_1\, k_2 \ldots k_\ell$

A toy key-related attack via bit tampering

- **Esempio:** lo stato segreto è una chiave $k \in \{0,1\}^\ell$. La funzionalità risponde con $\mathrm{Enc}_k(m)$ senza cambiare la chiave k, inoltre vale $\mathrm{Enc}_k(m) = \perp \iff \mathrm{parity}(k) = 1$.

execute(m)

$$\begin{cases} \neq \perp & \text{se } k_1 = 0 \\ \perp & \text{se } k_1 = 1 \end{cases}$$

Memoria: $0\, k_2 \ldots k_\ell$

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r, s)| = k/n \approx 1/5$ ed errore trascurabile.

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r,s)| = k/n \approx 1/5$ ed errore trascurabile.

Chi è l'avversario $\mathcal{F}_{\text{bit}}$?

È un avversario che manomette **ogni singolo bit** c_i in modo del tutto indipendente dagli altri:

$$f(c_1 \dots c_n) = f_1(c_1) \dots f_n(c_n)$$

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r,s)| = k/n \approx 1/5$ ed errore trascurabile.

Chi è l'avversario $\mathcal{F}_{\text{bit}}$?

È un avversario che manomette **ogni singolo bit** c_i in modo del tutto indipendente dagli altri:

$$f(c_1 \dots c_n) = f_1(c_1) \dots f_n(c_n)$$

Le uniche 4 operazioni possibili per ogni $f_i(x),\ x \in \{0,1\}$ sono:

Keep

$x \mapsto x$

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r,s)| = k/n \approx 1/5$ ed errore trascurabile.

Chi è l'avversario $\mathcal{F}_{\text{bit}}$?

È un avversario che manomette **ogni singolo bit** c_i in modo del tutto indipendente dagli altri:

$$f(c_1 \dots c_n) = f_1(c_1) \dots f_n(c_n)$$

Le uniche 4 operazioni possibili per ogni $f_i(x),\ x \in \{0,1\}$ sono:

Keep
$x \mapsto x$

Flip
$x \mapsto 1 - x$

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r,s)| = k/n \approx 1/5$ ed errore trascurabile.

Chi è l'avversario $\mathcal{F}_{\text{bit}}$?

È un avversario che manomette **ogni singolo bit** c_i in modo del tutto indipendente dagli altri:

$$f(c_1 \dots c_n) = f_1(c_1) \dots f_n(c_n)$$

Le uniche 4 operazioni possibili per ogni $f_i(x)$, $x \in \{0,1\}$ sono:

Keep	**Flip**	**Set0**
$x \mapsto x$	$x \mapsto 1 - x$	$x \mapsto 0$

Non-malleable codes per $\mathcal{F}_{\text{bit}}$

Teorema 4.1 (Dziembowski, Pietrzak, Wichs, *"Non-Malleable Codes"*, 2010)

Esiste una costruzione **esplicita** di un *Non-Malleable Code* $(\mathrm{Enc}, \mathrm{Dec})$ rispetto a $\mathcal{F}_{\text{bit}}$ con **rate costante** $|s|/|\mathrm{Enc}(r,s)| = k/n \approx 1/5$ ed errore trascurabile.

Chi è l'avversario $\mathcal{F}_{\text{bit}}$?

È un avversario che manomette **ogni singolo bit** c_i in modo del tutto indipendente dagli altri:

$$f(c_1 \dots c_n) = f_1(c_1) \dots f_n(c_n)$$

Le uniche 4 operazioni possibili per ogni $f_i(x)$, $x \in \{0,1\}$ sono:

Keep	**Flip**	**Set0**	**Set1**
$x \mapsto x$	$x \mapsto 1-x$	$x \mapsto 0$	$x \mapsto 1$

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

$\textsc{Execute}(m)$

Memoria:
$c = \mathrm{Enc}(s)$

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

$\textsc{Execute}(m)$

$s \leftarrow \mathrm{Dec}(c)$
$(s_1, a) \leftarrow G(s, m)$
$c_1 \leftarrow \mathrm{Enc}(s_1)$

Memoria:
$c = \mathrm{Enc}(s)$

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

$\textsc{Execute}(m)$

a

$s \leftarrow \mathrm{Dec}(c)$
$(s_1, a) \leftarrow G(s, m)$
$c_1 \leftarrow \mathrm{Enc}(s_1)$

Memoria:
$c_1 = \mathrm{Enc}(s_1)$

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

Memoria:
$c_1 = \mathrm{Enc}(s_1)$

Dai Non-Malleable Codes alla Tamper-Resilient security

L'Idea di base: invece di salvare lo stato s di una funzionalità G in chiaro, teniamo in memoria la sua codifica $c = \mathrm{Enc}(s)$.

Teorema 6.1 (Dziembowski, Pietrzak, Wichs - 2010)

Sia G una funzionalità e $(\mathrm{Enc}, \mathrm{Dec})$ un Non-Malleable Code rispetto a $\mathcal{F}$. Allora la funzionalità $G^{(\mathrm{Enc},\mathrm{Dec})}$ è **tamper-resilient** rispetto a $\mathcal{F}$:

il tampering sulla memoria attraverso $\mathcal{F}$ non dà all'avversario alcuna informazione aggiuntiva sullo stato iniziale.

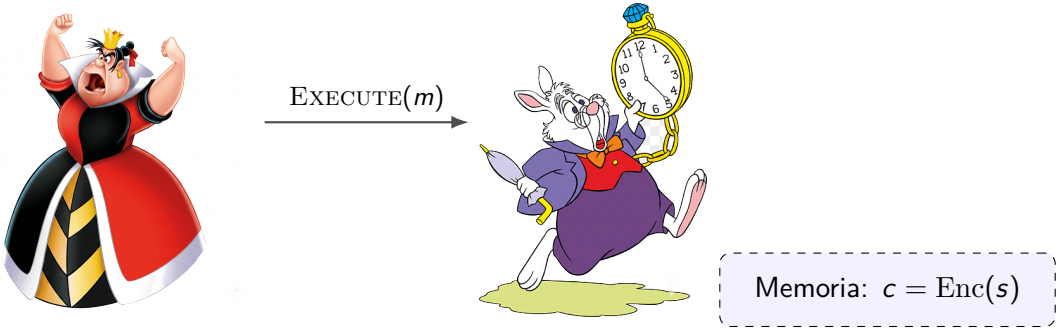
Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.

Memoria: $c = \mathrm{Enc}(s)$

Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.



Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.

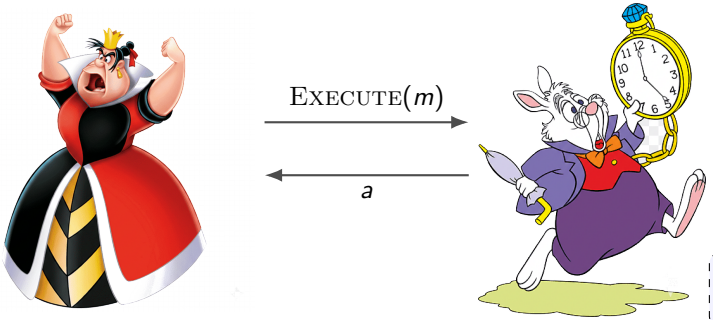
$\textsc{Execute}(m)$

$s \leftarrow \mathrm{Dec}(c)$
$(s_1, a) \leftarrow G(s, m)$
$c_1 \leftarrow \mathrm{Enc}(s_1)$

Memoria: $c = \mathrm{Enc}(s)$

Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.



Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.

$f \in \mathcal{F}$

Memoria: $c_1 = \mathrm{Enc}(s_1)$

Non-malleabilità $\Rightarrow$ il tampering induce o same? oppure $\tilde{s} \sim D_f$ (simulabile dato G pubblica).

Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.

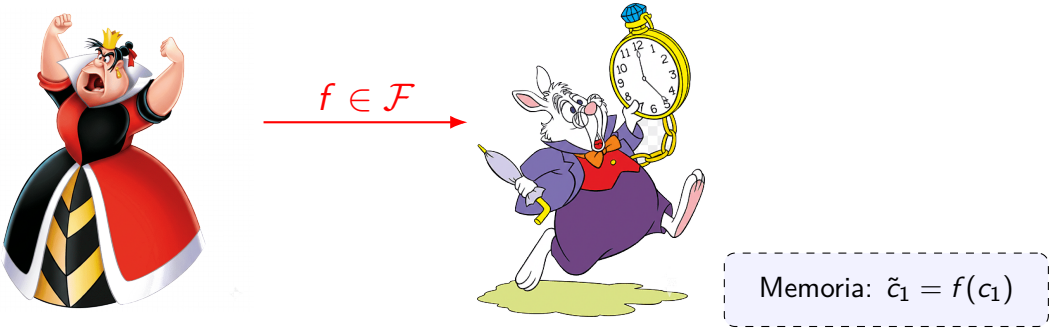
$f \in \mathcal{F}$

Memoria: $c_1 = \mathrm{Enc}(s_1)$

Non-malleabilità $\Rightarrow$ il tampering induce o same? oppure $\tilde{s} \sim D_f$ (simulabile dato G pubblica).

Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.



Non-malleabilità $\Rightarrow$ il tampering induce o same? oppure $\tilde{s} \sim D_f$ (simulabile dato G pubblica).

Idea della dimostrazione: hardening & simulazione

- **Hardening:** in memoria si salva $c = \mathrm{Enc}(s)$. Su $\textsc{Execute}(m)$: $s \leftarrow \mathrm{Dec}(c)$, $(s_1, a) \leftarrow G(s, m)$, $c_1 \leftarrow \mathrm{Enc}(s_1)$.

$f \in \mathcal{F}$

1. $\mathrm{Dec}(\tilde{c}_1) = s_1$ (same?)
2. $\mathrm{Dec}(\tilde{c}_1) = \tilde{s}_1,\ \tilde{s}_1 \leftarrow D_f$

Memoria: $\tilde{c}_1 = f(c_1)$

Non-malleabilità $\Rightarrow$ il tampering induce o same? oppure $\tilde{s} \sim D_f$ (simulabile dato G pubblica).

Il compromesso perfetto

- $\mathcal{F}_{\text{bit}}$ è un'ipotesi **troppo debole**: nella pratica l'avversario può fare molto di più che alterare i singoli bit indipendentemente.

Il compromesso perfetto

- $\mathcal{F}_{\text{bit}}$ è un'ipotesi **troppo debole**: nella pratica l'avversario può fare molto di più che alterare i singoli bit indipendentemente.
- Ma tollerare *tutte* le possibili manomissioni è matematicamente **impossibile**: non possono esistere non-malleable codes per $\mathcal{F} = \{\text{tutte le possibili funzioni } f : \{0,1\}^n \to \{0,1\}^n\}$.

Il compromesso perfetto: il modello Split-State

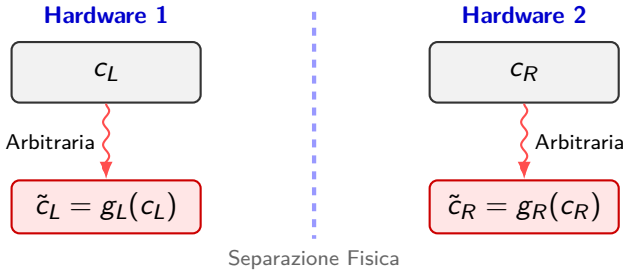
- $\mathcal{F}_{\text{bit}}$ è un'ipotesi **troppo debole**: nella pratica l'avversario può fare molto di più che alterare i singoli bit indipendentemente.
- Ma tollerare *tutte* le possibili manomissioni è matematicamente **impossibile**: non possono esistere non-malleable codes per $\mathcal{F} = \{\text{tutte le possibili funzioni } f : \{0,1\}^n \to \{0,1\}^n\}$.

La famiglia di funzioni $\mathcal{F}_{\text{split}}$

La *codeword* c viene divisa in due metà $c = (c_L, c_R)$, ognuna modificata arbitrariamente. Questa famiglia di funzioni è ben più vasta di $\mathcal{F}_{\text{bit}}$: ammettiamo funzioni g_L e g_R **complesse a piacere**, a patto che operino in modo indipendente sulle due metà:

$$\mathcal{F}_{\text{split}} = \Big\{ f(c_L, c_R) = \big(g_L(c_L)\,,\, g_R(c_R)\big) \;\Big|\; g_L, g_R \text{ funzioni arbitrarie} \Big\}$$

Il compromesso perfetto: il modello Split-State



La famiglia di funzioni $\mathcal{F}_{\text{split}}$

La *codeword* c viene divisa in due metà $c = (c_L, c_R)$, ognuna modificata arbitrariamente. Questa famiglia di funzioni è ben più vasta di $\mathcal{F}_{\text{bit}}$: ammettiamo funzioni g_L e g_R **complesse a piacere**, a patto che operino in modo indipendente sulle due metà:

$$\mathcal{F}_{\text{split}} = \Big\{ f(c_L, c_R) = \big(g_L(c_L)\,,\, g_R(c_R)\big) \;\Big|\; g_L, g_R \text{ funzioni arbitrarie} \Big\}$$

Dagli Expander Graphs ai Non-Malleable Codes (1 bit)

Graph Code

Dato un grafo $G = (V, E)$ con $|V| = 2^k$:

Dagli Expander Graphs ai Non-Malleable Codes (1 bit)

Graph Code

Dato un grafo $G = (V, E)$ con $|V| = 2^k$:

Codifica 1

$\mathrm{enc}_G(1) = (u, v) \leftarrow E$

(arco casuale)

Dagli Expander Graphs ai Non-Malleable Codes (1 bit)

Graph Code

Dato un grafo $G = (V, E)$ con $|V| = 2^k$:

Codifica 1

$\mathrm{enc}_G(1) = (u, v) \leftarrow E$

(arco casuale)

Codifica di 0

$\mathrm{enc}_G(0) = (u, v) \leftarrow (V \times V) \setminus E$

(non-arco casuale)

Dagli Expander Graphs ai Non-Malleable Codes (1 bit)

Graph Code

Dato un grafo $G = (V, E)$ con $|V| = 2^k$:

Codifica 1

$\mathrm{enc}_G(1) = (u, v) \leftarrow E$

(arco casuale)

Codifica di 0

$\mathrm{enc}_G(0) = (u, v) \leftarrow (V \times V) \setminus E$

(non-arco casuale)

Decodifica

$\mathrm{dec}_G(u, v) = \mathbb{1}[(u, v) \in E]$

(1 se arco, 0 altrimenti)

Dagli Expander Graphs ai Non-Malleable Codes (1 bit)

Graph Code

Dato un grafo $G = (V, E)$ con $|V| = 2^k$:

Codifica 1	**Codifica di 0**	**Decodifica**
$\mathrm{enc}_G(1) = (u, v) \leftarrow E$	$\mathrm{enc}_G(0) = (u, v) \leftarrow (V \times V) \setminus E$	$\mathrm{dec}_G(u, v) = \mathbb{1}[(u, v) \in E]$
(arco casuale)	(non-arco casuale)	(1 se arco, 0 altrimenti)

Teorema 6 (Rasmussen–Sahai, *"Expander Graphs are Non-Malleable Codes"*, 2019)

Se G è d-regolare con espansione spettrale σ_2 e $n = \Omega(d^3 \log d/\lambda)$, allora $(\mathrm{enc}_G, \mathrm{dec}_G)$ è un ε-NMC nello split-state model con

$$\varepsilon = O\left(\frac{\sigma_2^{3/2}}{d}\right)$$

e la dimostrazione si basa su **due applicazioni annidate** dell'Expander Mixing Lemma!