Random Oracle Model

Paolo D'Arco

Difficoltá di esibire prove

Molte costruzioni basate su funzioni hash crittografiche **non** possono essere provate sicure usando soltanto assunzioni tipo

- la funzione hash é collision-resistant
- la funzione hash é second-preimage resistant
- la funzione hash é pre-image resistant

Come comportarsi? Che fare?

Individuare e scrutinare nuove assunzioni ragionevoli sulle funzioni hash o sulla specifica funzione hash usata nella costruzioni. Puó richiedere tempo.

Schemi per cui é possibile esibire una prova sono spesso molto *meno efficienti* di schemi che usano funzioni hash per cui una prova non c'é.

Random Oracle Model

D'altra parte, l'uso di schemi basato soltanto sulla giustificazione che sono efficienti e tutti gli attacchi noti falliscono é contro l'approccio rigoroso discusso sino ad ora.

Un approccio che ha avuto un notevole successo consiste nell'usare un **modello idealizzato** per derivare una misura di confidenza nella robustezza della costruzione progettata.

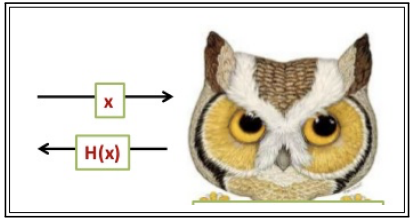
L'esempio piú significativo é il **Random Oracle Model**.

Nel modello si assume la presenza di

$$H : \{0,1\}^* \to \{0,1\}^\ell \qquad \text{funzione } \textbf{totalmente casuale}$$

pubblica, che puó essere valutata soltanto attraverso query ad un oracolo.

Random Oracle Model



Tutte le parti in uno schema crittografico, partecipanti onesti **e** avversario, possono interagire con il random oracle

Nota: un random oracle **non** esiste nella realtá. É una astrazione. Memorizzare una funzione casuale richiede spazio esponenziale.

Random Oracle Model

Passi della metodologia

1. Uno schema viene progettato e provato sicuro nel modello in cui é presente il random oracle
2. Per implementare lo schema nel mondo reale, **istanziamo** il random oracle con una funzione hash crittografica $\hat{H}$

Quando lo schema impone ad una delle parti di **inviare** una query all'oracolo, nell'implementazione la parte **calcola** l'output della funzione hash $\hat{H}$

Se $\hat{H}$ é **sufficientemente buona** nell'emulare il random oracle, la prova di sicurezza data nel modello **dovrebbe** estendersi allo schema reale.

Random Oracle Model

Sfortunatamente:

- non c'é nessuna giustificazione teorica per la precedente affermazione
- ci sono esempi (artificiosi...) di schemi sicuri nel random oracle model, che risultano insicuri per **qualsiasi** istanziazione concreta con $\hat{H}$
- non é chiaro cosa significhi "sufficientemente buona" nell'emulare il random oracle

$\hat{H}$ é deterministica e fissata

$\Downarrow$

non puó comportarsi come una funzione totalmente casuale

Random Oracle Model

Analizziamo la metodologia nel dettaglio

- le parti oneste e Adv possono inviare query x all'oracolo ricevendo $H(x)$
- i meccanismi interni dell'oracolo non sono noti (scatola nera)
- le query sono **private**
  - nessun altro conosce la query x fatta da una parte
  - non sa neanche che ha inviato una query
- le query vengono risposte **consistentemente**, i.e., stessa query - stessa risposta

Random Oracle Model

Ricordo che la scelta di una funzione casuale puó essere effettuata in due modi

1. *one-shot*: scegliere in un solo colpo nell'insieme di tutte le funzioni su dominio e codominio specificati
2. *on-the-fly*: scegliere uniformemente al volo le risposte alle query e memorizzarle in tabella

La seconda opzione é piú facile da gestire ed anche piú appropriata quando H é definita su un dominio infinito

Random Oracle Model

Osservazione: abbiamo giá usato funzioni casuali in precedenza ma **in modo diverso**.

Per definire funzioni pseudocasuali: in quel caso sono usate peró come **termine di paragone**

$$\Downarrow$$

cioé come un modo per definire **cosa significa** essere pseudocasuale

Nel random oracle model, la funzione random H é **parte della costruzione stessa**

$$\Downarrow$$

in una implementazione deve essere istanziata con una $\hat{H}$

Random Oracle Model

Le definizioni che abbiamo considerato fino ad ora nel cosiddetto *modello standard* (per distinguerlo dal *random oracle model*) hanno la forma:

Π é sicura se, $\forall$ A ppt, la probabilitá di un certo evento é sotto una determinata soglia, dove la probabilitá é calcolata sulle scelte casuali

- delle parti oneste
- dell'avversario A

Nel random oracle model (ROM, per brevitá), Π poggia su un oracolo:

Π é sicura se, $\forall$ A ppt, la probabilitá di un certo evento é sotto una determinata soglia, dove la probabilitá é calcolata sulle scelte casuali

- delle parti oneste
- dell'avversario A
- **dell'oracolo H**

Prove nel ROM

Le dimostrazioni nel modello ROM possono sfruttare il fatto che H viene scelta uniformemente a caso e puó essere valutata soltanto tramite query.

Tre proprietá risultano utili:

- *uniformitá:* se x non é stato ancora usato per una query ad H, allora il valore $H(x)$ é uniformemente distribuito

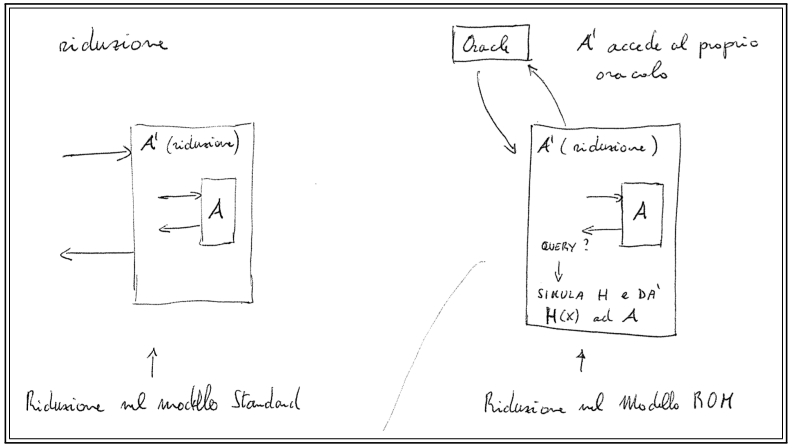
Si noti la differenza tra H ed un PRG

$G(x)$ é pseudocasuale per un osservatore ppt, assumendo che x sia scelto uniformemente e sia totalmente sconosciuto all'osservatore.

$H(x)$ é *totalmente casuale* fino a quando un osservatore non ha inviato la query x all'oracolo. **Anche** se x é noto o se x non é scelto uniformemente.

Prove nel ROM

Le altre due proprietá sono relative alle prove di sicurezza:



Prove nel ROM

La riduzione A' nel ROM deve simulare l'oracolo H con cui interagisce A, rispondendo opportunamente alle sue query.

- *estraibilitá:* se A invia la query x ad H, la riduzione A' *vede* questa query ed *acquisisce* il valore di x
- *programmabilitá:* la riduzione A' puó fissare il valore $H(x)$ ad un valore di propria scelta, posto che il valore sia uniformemente distribuito

Osservazione: quando H, in una implementazione, viene istanziata con una funzione concreta $\hat{H}$, **non** c'é nessuna controparte per le proprietá dell'estraibilitá e della programmabilitá

$$\Downarrow$$

Lo schema "concreto" diverge dallo schema nel modello e la prova di sicurezza usa possibilitá che **non** esistono nell'analisi dello schema concreto.

ROM: Metodologia robusta?

Cosa garantisce una riduzione nel modello ROM?

Il dibattito é acceso nella comunitá.

Obiezioni al ROM:

- Non c'é prova o giustificazione rigorosa che $\hat{H}$ invece dell'oracolo H mantiene lo schema sicuro nel mondo reale come fa H nel ROM
- Nessun $\hat{H}$ puó comportarsi come un oracolo casuale. Un Adv acquisisce al **descrizione** di $\hat{H}$. Gli output sono determinati univocamente.
- Non sappiamo cosa significa che $\hat{H}$ é sufficientemente buona nell'emulare H
- Funzioni hash concrete, e.g., SHA2, possono andar bene in alcune applicazioni non in tutte.

ROM: Metodologia robusta?

Si noti che esiste una differenza qualitativa nelle assunzioni.

1. SHA-2 si comporta come un oracolo casuale
2. SHA-2 é collision resistant
3. AES é una pseudorandom function

Per la prima non c'é una definizione soddisfacente di cosa significhi. Per la seconda e la terza esistono delle buone definizioni di queste proprietá.

$\Rightarrow$ l'assunzione *ROM* é qualitativamente differente da assunzioni "ben definite".

ROM: Metodologia robusta?

Motivazioni a supporto del ROM (che spiegano perché é stato usato ampiamente)

1. Rende possibile la progettazione di protocolli spesso piú **efficienti** di quelli realizzati nel modello standard
2. Le prove nel ROM sono quasi universalmente accettate/riconosciute come elementi che accrescono la **confidenza** nella sicurezza di uno schema
3. Una prova offre garanzie sulla **soliditá** del progetto. Soltanto attacchi nel mondo reale che sfruttano le debolezze di $\hat{H}$ possono avere successo
4. Ad oggi **non** ci sono stati attacchi che hanno avuto successo nel mondo reale su schemi che sono stati provati sicuri nel ROM, quando H é stato istanziato propriamente