Costruzioni sicure

Paolo D'Arco
pdarco@unisa.it

Università di Salerno

Elementi di Crittografia

Contenuti

Costruzioni di schemi di cifratura sicuri

Abbiamo bisogno di blocchi di base:

- Generatori pseudocasuali (pseudorandom generator, PRG)

Un generatore pseudocasuale G è un algoritmo deterministico efficiente per trasformare una stringa *uniforme* corta, chiamata *seme*, in una più lunga che *sembra uniforme*

Studiati a partire dagli anni '40

- progettati per superare test statistici ad hoc: primo bit, parità, ...
- nessuna garanzia di robustezza in specifiche applicazioni

Generatori pseudocasuali: approccio

Approccio crittografico negli anni '80

- un generatore pseudocasuale buono dovrebbe superare *tutti i test statistici efficienti*
- ad ogni *osservatore efficiente* l'output dovrebbe sembrare una stringa uniforme

Nota: usiamo le espressioni *stringa pseudorandom/pseudocasuale* e *stringa uniforme* con abuso della terminologia.

- pseudocasualità: proprietà di una distribuzione di stringhe
- uniformità: proprietà di una distribuzione di stringhe

Distribuzione pseudocasuale

Che cosa significa per una distribuzione di probabilità essere *pseudocasuale*?

Sia *Dist* una distribuzione su stringhe di ℓ bit. È pseudocasuale

- se l'esperimento in cui una stringa viene campionata in accordo a *Dist* è *indistinguibile* dall'esperimento in cui una stringa di lunghezza ℓ viene campionata in accordo alla distribuzione uniforme

Risulta, cioè, impossibile per ogni algoritmo PPT dire, con chance significativamente migliori di quelle offerte dal lancio di una moneta, se essa derivi da *Dist* o dalla distribuzione uniforme

Discende che *una stringa pseudocasuale è tanto buona quanto una uniforme.*

Generatore pseudocasuale (informale)

Sia $G : \{0,1\}^n \to \{0,1\}^\ell$ e sia *Dist* la distribuzione sulle stringhe di ℓ bit ottenuta

- scegliendo uniformemente a caso $s \in \{0,1\}^n$
- dando in output $G(s)$

G è PRG se e solo se *Dist* è pseudocasuale.

Generatore pseudocasuale (formale)

Definizione 3.14. Sia $\ell(n)$ un polinomio e G un algoritmo deterministico di tempo polinomiale tale che, per ogni n ed $s \in \{0,1\}^n$, $G(s)$ è una stringa di $\ell(n)$ bit. Diremo che G è un PRG se valgono le seguenti condizioni:

1. **Espansione**: per ogni n risulta $\ell(n) > n$
2. **Pseudocasualità**: per ogni algoritmo D PPT, esiste una funzione trascurabile negl(n) tale che

$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq negl(n)$$

dove la prima probabilità è calcolata su

- scelta uniforme di $s \in \{0,1\}^n$
- random bit di D

e la seconda su

- scelta uniforme di $r \in \{0,1\}^{\ell(n)}$
- random bit di D

Una costruzione non sicura

Esempio 3.15. Sia $G : \{0,1\}^n \to \{0,1\}^{n+1}$ tale che $G(s) = s || \bigoplus_{i=1}^{n} s_i$.

Distinguisher D:

su input ω dà in output 1 se e solo se il bit finale di ω è l' *xor* di tutti i precedenti.

Risulta:

- se l'input di D è $G(s)$, allora $Pr[D(G(s)) = 1] = 1$
- se l'input di D è r, allora $Pr[D(r) = 1] = 1/2$
  - la probabilità che l'ultimo bit sia uguale all'xor dei precedenti è $1/2$ perchè r viene scelta in modo uniforme in $\{0,1\}^{n+1}$

Poichè la differenza è

$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| = |1 - 1/2| = 1/2,$$

costante (non trascurabile), G non è pseudocasuale.

Osservazioni

La distribuzione sugli output di un generatore pseudocasuale G è lontana da quella uniforme. Per rendersene conto, sia $\ell(n) = 2n$. Allora

$$G : \{0,1\}^n \rightarrow \{0,1\}^{2n}.$$

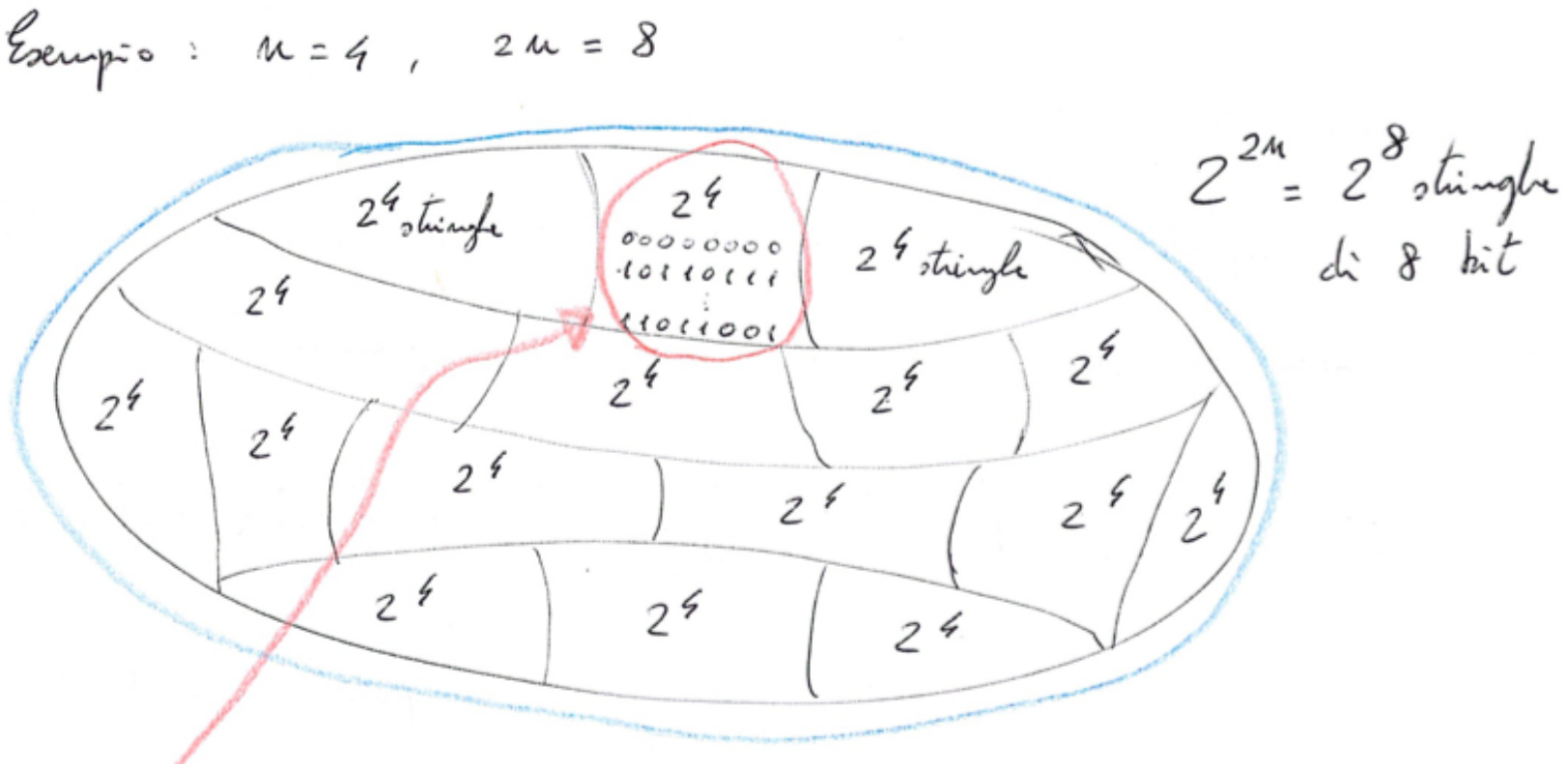
Poichè ci sono 2^n possibili semi, G può generare *al più* 2^n delle 2^{2n} possibili stringhe.

Equivalentemente, la frazione di stringhe di $2n$ bit che G può generare è

$$2^n/2^{2n} = 1/2^n.$$

Si consideri il seguente esempio, in cui $n = 4$

Osservazioni



Nell'esempio G può generare soltanto $2^4 = 16$ stringhe di 8 bit, dell'universo delle possibili $2^8 = 256$ stringhe di 8 bit. La *stragrande maggioranza* non può essere generata.

Osservazioni

Data una quantità illimitata di tempo è pertanto banale distinguere.

Un distinguisher D di **tempo esponenziale** tenta tutti i semi $s \in \{0,1\}^n$ e dà in output 1 se e solo se $G(s) = \omega$ per qualche s.

Si noti che, presa una stringa ω uniforme in $\{0,1\}^{2n}$, risulta $Pr[G(s) = \omega] \leq 1/2^n$. Pertanto

$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| = 1 - 1/2^n \text{ non trascurabile.}$$

Quindi D distingue attraverso una ricerca esaustiva del seme in $s \in \{0,1\}^n$.

Ciò **non** contraddistingue la pseudocasualità di G perchè D **non** è efficiente computazionalmente.

Osservazioni

Il seme di G equivale alla chiave in uno schema di cifratura simmetrico

- deve essere scelto in modo uniforme e tenuto segreto
- abbastanza lungo da evitare ricerche esaustive

La lunghezza del seme equivale al parametro di sicurezza.

> Domanda: ma esistono generatori pseudocasuali?

Nessuna prova *incondizionata*, prova *sotto assunzioni*.

Se esistono funzioni *one-way* (a senso unico), allora esistono PRG (... ci torneremo).

In pratica abbiamo buone costruzioni che utilizzano stream cipher.

Stream cipher

La definizione di PRG ha due limitazioni

- il fattore di espansione è fisso
- l'intero output viene prodotto tutto d'un colpo

Al contrario, uno stream cipher

- non ha un limite al fattore di espansione
- produce bit di output gradualmente

Stream cipher: coppia di algoritmi $(Init, GetBits)$ deterministici, dove

- $Init(s, IV)$ dà in output uno stato iniziale s_0
- $GetBits(st_{i-1})$ dà in output (st_i, y) (prox stato, bit/stringa di bit)

PRG da uno stream cipher

Dato uno stream cipher ed un fattore di espansione ℓ, possiamo definire $G_\ell : \{0,1\}^n \to \{0,1\}^\ell$ come segue:

ALGORITHM 3.16
Constructing G_ℓ from (Init, GetBits)

Input: Seed s and optional initialization vector IV
Output: $y_1, \ldots, y_\ell$

$\mathsf{st}_0 := \mathsf{Init}(s, IV)$
for $i = 1$ to ℓ:
    $(y_i, \mathsf{st}_i) := \mathsf{GetBits}(\mathsf{st}_{i-1})$
return $y_1, \ldots, y_\ell$

Sicurezza di uno stream cipher

Uno stream cipher è sicuro (in un senso di base) se non prende in input IV e, per ogni polinomio $\ell(n) > n$, la funzione G_ℓ è un generatore pseudocasuale con fattore di espansione ℓ.

In altre parole, lo stream cipher è sicuro se G_ℓ è un PRG

Nelle prossime lezioni vedremo alcune costruzioni usate in pratica di stream cipher.

Prove (dimostrazioni) per riduzione

Provare che una costruzione è computazionalmente sicura significa

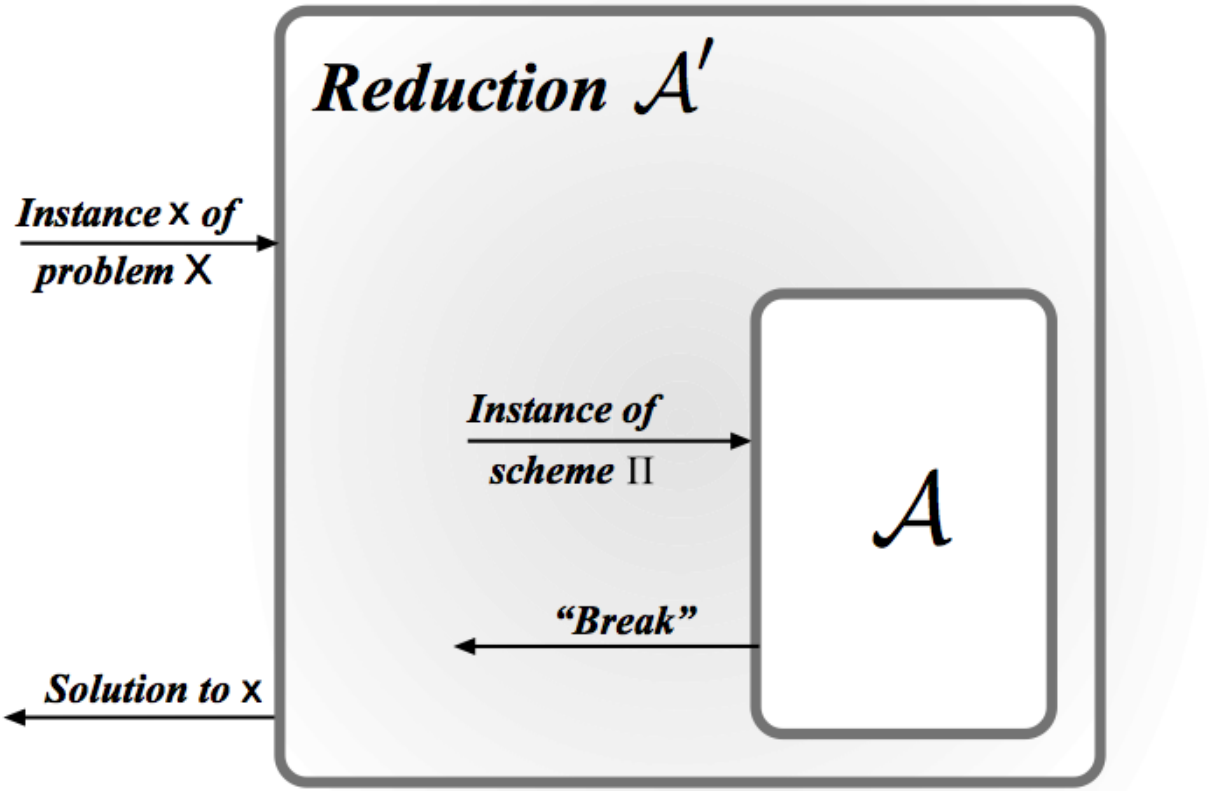
- far affidamento su assunzioni (aff. non provate, congetturate vere)
  - un problema matematico è difficile
  - qualche primitiva crittografica è sicura
- provare che, basandosi sulle assunzioni fatte, la costruzione risulta sicura.

Generalmente procederemo come segue:

Presenteremo una riduzione esplicita che mostra come trasformare un Adv efficiente A che ha successo nel rompere la costruzione data in un Adv efficiente A' che

- risolve il problema matematico supposto difficile
- rompe la primitiva crittografica assunta sicura

Struttura di una riduzione



Cominciamo con una assunzione: il problema X è difficile, i.e., non può essere risolto con algoritmi PPT.

Vogliamo provare che la costruzione Π è sicura

Struttura di una riduzione

1. Fissiamo un Adv A PPT che attacca Π con probabilità di successo $\epsilon(n)$
2. Costruiamo A' PPT (detto la *riduzione*) che risolve X usando A come subroutine. A' non conosce il funzionamento interno di A; sa solo che serve ad attaccare Π.

   Quindi, data x, un'istanza di X, A' simula per A un'istanza della costruzione Π in modo tale che:

   1. A pensa di star interagendo con Π, i.e., ha la *stessa vista* (o molto simile) che ha quando interagisce realmente con Π
   2. se A ha successo nel rompere Π sull'istanza che ha ricevuto da A', allora A' risolve l'istanza x del problema X che ha ricevuto in input con probabilità almeno $1/p(n)$

Struttura di una riduzione

3. Le condizioni 2.1 e 2.2 implicano che
   - A' risolve X con probabilità almeno $\epsilon(n)/p(n)$. Pertanto, se $\epsilon(n)$ è non trascurabile, anche $\epsilon(n)/p(n)$ è non trascurabile
   - A' è efficiente se A è efficiente.

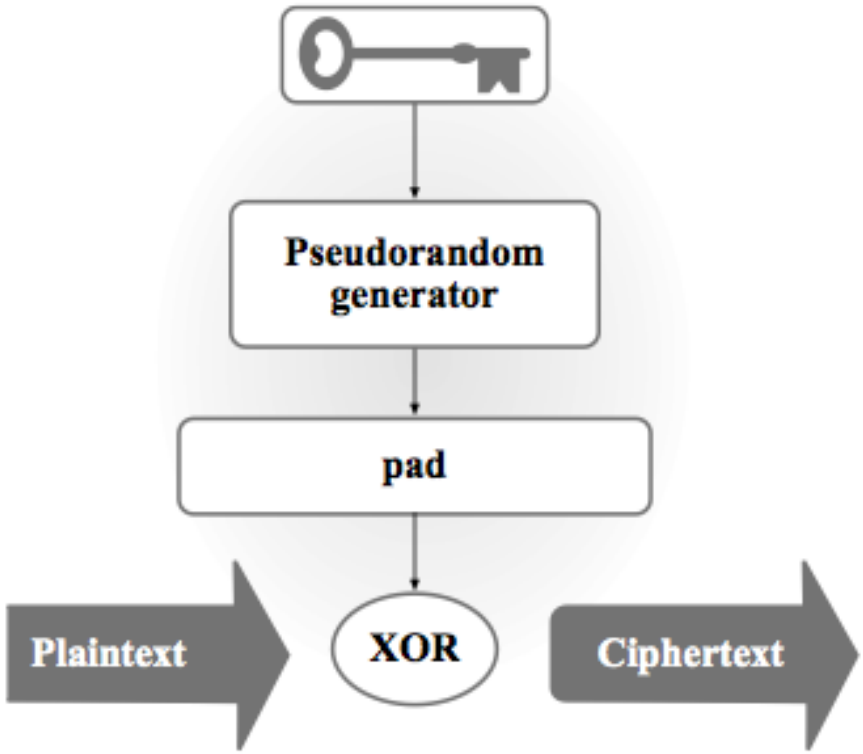
4. Data l'assunzione di difficoltà sul problema X, concludiamo che *nessun* Adv A PPT può avere successo nel rompere Π con probabilità non trascurabile. Quindi, Π è computazionalmente sicuro.

Costruiremo ora uno schema di cifratura utilizzando un PRG ed esemplificheremo la tecnica provandone la sicurezza.

Cifratura sicura con un PRG per messaggi di lunghezza fissa

Idea:

- il one-time pad cifra tramite $m \oplus r$, con r stringa scelta *uniformemente a caso*
- qui invece cifriamo tramite $m \oplus G(s)$, con $G(s)$ stringa *pseudocasuale*
- le parti devono condividere il seme (chiave segreta)



Cifratura sicura con un PRG

CONSTRUCTION 3.17

Let G be a pseudorandom generator with expansion factor ℓ. Define a private-key encryption scheme for messages of length ℓ as follows:

- Gen: on input 1^n, choose uniform $k \in \{0,1\}^n$ and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the message
$$m := G(k) \oplus c.$$

Per ogni $k \in \{0,1\}^n$ e per ogni $m \in \{0,1\}^\ell$, risulta:

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

Cifratura sicura con un PRG

Teorema 3.18 Se G è un PRG, la Costruzione 3.17 realizza uno schema di cifratura a chiave privata per messaggi di lunghezza fissa che ha cifrati indistinguibili in presenza di un eavesdropper.

Dim. Facciamo vedere che, per ogni Adv A PPT, esiste una fun. tras. $negl(n)$ tale che

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq 1/2 + negl(n).$$

Intuizione della prova:

- se Π usasse un pad uniforme invece di $G(k)$, Π sarebbe identico allo schema one-time pad ed A vincerebbe nell'esperimento con prob. $1/2$
- pertanto, se A fosse in grado di vincere nel nostro caso con probabilità significativamente maggiore di $1/2$, allora A potrebbe essere usato per *distinguere* $G(k)$ da una stringa uniforme

Cifratura sicura con un PRG

Procediamo formalmente, esibendo la riduzione.

Costruiamo D che distingue $G(k)$ da r, utilizzando A e la sua abilità nel capire quale messaggio tra m_0 ed m_1 è stato cifrato

Probabilità di successo di D relata alla probabilità di successo di A.

Distinguisher D:

Riceve in input una stringa $\omega \in \{0,1\}^{\ell(n)}$

1. esegue $A(1^n)$ per ottenere $m_0, m_1 \in \{0,1\}^{\ell(n)}$
2. sceglie uniformemente $b \in \{0,1\}$ e pone $c := m_b \oplus \omega$
3. dà c ad A ed ottiene da A il bit b'
4. se $b' = b$, dà in output 1; altrimenti, dà in output 0

Cifratura sicura con un PRG

Osservazioni:

- D emula l'esperimento $PrivK^{eav}_{A,\Pi}(n)$ per A ed osserva se A ha successo o meno. Nel primo caso, D pensa che ω debba essere pseudocasuale; nel secondo, uniforme.
- D è PPT se A è un Adv PPT.

Al fine di analizzare D, definiamo lo schema $\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$.

È esattamente il one-time pad: $\widetilde{Gen}\,(1^n)$ riceve in input il parametro di sicurezza n e dà in output una chiave *uniforme* $k \in \{0,1\}^{\ell(n)}$

La segretezza perfetta di $\widetilde{\Pi} \quad \Rightarrow \quad Pr[PrivK^{eav}_{A,\widetilde{\Pi}}(n) = 1] = 1/2.$

Cifratura sicura con un PRG

Ora, segue che:

1. se ω è una stringa uniforme in $\{0,1\}^{\ell(n)}$ allora
   - la vista di A quando eseguito come subroutine da D = la vista di A in $PrivK^{eav}_{A,\widetilde{\Pi}}(n)$
   - poichè D dà in output 1 quando A ha successo, risulta

$$Pr_{\omega \leftarrow \{0,1\}^{\ell(n)}}[D(\omega) = 1] = Pr[PrivK^{eav}_{A,\widetilde{\Pi}}(n) = 1] = 1/2.$$

2. se, invece, $\omega = G(k)$, dove k è una stringa uniforme in $\{0,1\}^n$ allora
   - la vista di A quando eseguito come subroutine da D = la vista di A in $PrivK^{eav}_{A,\Pi}(n)$
   - poichè D dà in output 1 quando A ha successo, risulta

$$Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1] = Pr[PrivK^{eav}_{A,\Pi}(n) = 1].$$

Cifratura sicura con un PRG

Se risultasse $Pr[PrivK^{eav}_{A,\Pi}(n) = 1] = 1/2 + \textit{non-negl}(n)$ allora

$$\begin{aligned} &|Pr[PrivK^{eav}_{A,\widetilde{\Pi}}(n) = 1] - Pr[PrivK^{eav}_{A,\Pi}(n) = 1]| \\ &= |1/2 - (1/2 + \textit{non-negl}(n))| = \textit{non-negl}(n) \end{aligned}$$

cioé, per le precedenti,

$$|Pr_{\omega \leftarrow \{0,1\}^{\ell(n)}}[D(\omega) = 1] - Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1]| = \textit{non-negl}(n).$$

Dunque D é PPT e distingue con probabilitá $\textit{non-negl}(n)$! Ma G per ipotesi è un PRG. Pertanto, non puó essere. Deve esistere $\textit{negl}(n)$ tale che

$$|Pr_{\omega \leftarrow \{0,1\}^{\ell(n)}}[D(\omega) = 1] - Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1]| \leq \textit{negl}(n)$$

che implica

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq 1/2 + \textit{negl}(n).$$

△

Semplificazione della prova: alternativa alla slide precedente

G per ipotesi è un PRG e D è PPT. Pertanto, esiste $negl(n)$ tale che

$$|Pr_{\omega \leftarrow \{0,1\}^{\ell(n)}}[D(\omega) = 1] - Pr_{k \leftarrow \{0,1\}^{n}}[D(G(k)) = 1]| \leq negl(n).$$

Ma allora,

$$|Pr[PrivK^{eav}_{A,\widetilde{\Pi}}(n) = 1] - Pr[PrivK^{eav}_{A,\Pi}(n) = 1]| \leq negl(n),$$

ovvero

$$|1/2 - Pr[PrivK^{eav}_{A,\Pi}(n) = 1]| \leq negl(n),$$

che implica

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq 1/2 + negl(n).$$

Poichè A è un Adv PPT arbitrario, possiamo concludere che Π ha cifrati indistinguibili in presenza di un eavesdropper. $\triangle$

Commenti

Commenti:

- la dimostrazione *non* è incondizionata. Π è sicuro se G è un PRG
- l'approccio che consiste nel basare la sicurezza di costruzioni di alto livello su costruzioni di basso livello ha diversi vantaggi
  - è più facile progettare primitive di basso livello
  - è più facile l'analisi rispetto ad una definizione più semplice
  - una buona primitiva può essere usata in molte costruzioni di alto livello

Quale vantaggio abbiamo rispetto al one-time pad?

Una chiave k di pochi bit, e.g., 128, permette di cifrare messaggi molto lunghi, e.g., di 1 GB

Limitazione concreta

Come derivare una limitazione concreta alla sicurezza di Π usando una limitazione concreta alla sicurezza di G?

Fissiamo n e assumiamo che G sia (t, ϵ)-pseudocasuale per questo n.

$\Rightarrow \forall D$ che esegue in tempo al più t,

$$|Pr[D(r) = 1] - Pr[D(G(s)) = 1]| \leq \epsilon.$$

Per esempio, potremmo avere $t \approx 2^{80}$ ed $\epsilon \approx 2^{-60}$.

Possiamo far vedere che Π è $(t - c, \epsilon)$-EAV-sicuro per qualche costante c piccola, i.e., $\forall A$ che esegue in tempo al più $t - c$, risulta

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon.$$

Limitazione concreta

Infatti, sia A un avversario arbitrario di tempo al più $t - c$. Il distinguisher D che abbiamo costruito *fa ben poco* oltre ad invocare A ed eseguirlo.

Ponendo c al valore opportuno per tener conto del lavoro aggiuntivo di D, risulta il tempo totale di esecuzione di D al più $(t - c) + c = t$.

L'assunzione che G sia un PRG implica che

$$|Pr[D(r) = 1] - Pr[D(G(s)) = 1]| \leq \epsilon$$

e, quindi, ripetendo gli stessi passi della riduzione, si ottiene che

$$Pr[PrivK^{eav}_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon.$$

Nota che stiamo parlando di valori *concreti*. Il parametro di sicurezza n è fissato all'inizio e la valutazione è per valori specifici, *non asintotica*.