Approccio computazionale

Pseudorandomness

Paolo D’Arco

Cifratura simmetrica

La segretezza perfetta ha limitazioni intrinseche

La crittografia moderna nasce dall'esigenza di superarle

Concetti chiave:

- Segretezza **computazionale**: una chiave puó essere usata per cifrare molti messaggi lunghi, e.g., chiave di 128 bit per diversi GigaByte di dati
  - Piú debole di quella perfetta ma ... sufficiente
- **Pseudocasualitá** (pseudorandomness): cattura l'idea che un oggetto puó sembrare completamente casuale anche se non lo é
  - Centrale per lo sviluppo di tutta la crittografia moderna, con implicazioni in altri campi

Cifratura simmetrica

Piú in dettaglio, la segretezza perfetta garantisce

- assolutamente nessuna informazione per l'Adv che ascolta
- Adv ha potere illimitato

Da un punto di vista pratico uno schema che

- rivela informazioni con probabilitá 2^{-60}
- ad Adv che possono investire 200 anni di sforzo computazionale

é ancora molto buono.

Cifratura simmetrica

Le definizioni di sicurezza che tengono di conto

- i limiti computazionali dell'Adv
- ammettono una piccola probabilitá di errore

vengono dette **computazionali** (per distinguerle da quelle basate sulla teoria dell'Informazione)

In questo caso

- la sicurezza é soltanto garantita rispetto ad Adv efficienti
- Adv possono potenzialmente avere successo ma soltanto con probabilitá molto piccola.

Cifratura simmetrica

Discende che

- Le risorse richieste per "rompere" uno schema devono essere maggiori di quelle di cui Adv puó disporre
- Le probabilitá devono essere realmente piccole

Ci sono due approcci per sviluppare una teoria significativa

- Concreto
- Asintotico

Approccio concreto

Limita esplicitamente la probabilitá di successo massima di qualsiasi Adv probabilistico che esegue per una specificata quantitá di tempo

> *Uno schema é (t, ϵ)-sicuro se qualsiasi Adv che puó eseguire per tempo al piú t ha successo nella rottura dello schema con probabilitá al piú ϵ.*

Parametri reali potrebbero essere oggi: $t = 2^{80}$ cicli di CPU, $\epsilon = 2^{-60}$.

Esempio 3.1. I moderni schemi di cifratura simmetrici sono considerati quasi ottimali se, quando la chiave é lunga n bit (spazio delle chiavi 2^n elementi) qualsiasi Adv che esegue per tempo t ha successo nella rottura dello schema con probabilitá al piú $\frac{ct}{2^n}$, per qualche costante c fissata.

Approccio concreto

La condizione precedente corrisponde ad un attacco di forza bruta

Essenzialmente dice che non ci sono attacchi migliori della ricerca esaustiva

Assumiamo $c = 1$, lunghezza chiave 60 bit ed un PC che esegue 4×10^9 cicli per secondo. La ricerca esaustiva richiede:

$$\frac{2^{60}}{4 \times 10^9} \text{ secondi, che corrispondono a circa 9 anni}$$

Rispetto ad un computer molto piú potente che puó effettuare 2×10^{16} cicli per sec

$$\frac{2^{60}}{2 \times 10^{16}} \text{ secondi, che corrispondono a circa 1 minuto.}$$

Con lo stesso computer, per una chiave lunga 80 bit

$$\frac{2^{80}}{2 \times 10^{16}} \text{ secondi, che corrispondono a circa 2 anni.}$$

La lunghezza raccomandata oggi é $n = 128$ bit.

Approccio concreto

Per avere un'idea di quanto siano grandi questi numeri ...

- 2^{128} é la cardinalitá dello spazio delle chiavi
- $2^{58} \approx$ il numero di secondi di vita dell'Universo dal Big Bang ad oggi

E per avere un'idea di quanto siano piccole le probabilitá ...

Se un Adv ha successo in un anno di tempo con prob. 2^{-60}*, é molto piú probabile che Alice e Bob siano colpiti entrambi da un fulmine nello stesso periodo di tempo*

Svantaggi dell'approccio concreto

Difficoltá di interpretazione di un'affermazione.

Nessun Adv che esegue per 5 *anni ha successo con prob. maggiore di* ϵ

- con quanto potere computazionale?
- tiene conto di sviluppi futuri (legge di Moore)?
- considera implementazioni generiche o ad hoc degli attacchi?
- cosa possiamo dire di un Adv che esegue per 2 anni?
- cosa di un Adv che esegue per 10 anni?

Approccio asintotico

Introduce un *parametro di sicurezza* n intero

- Parametrizza sia lo schema che le parti coinvolte: partecipanti, Adv
- Puó essere pensato al momento come la lunghezza della chiave
- É noto ad Adv

I tempi di esecuzione di Adv, dei partecipanti e la prob. di successo sono funzioni di n.

- Avversari efficienti: algoritmi probabilistici di tempo polinomiale in n
- Partecipanti: algoritmi probabilistici di tempo polinomiale in n
- Probabilitá di successo piccola: probabilitá piú piccola dell'inverso di *ogni* polinomio in n, i.e., $1/p(n)$, dette probabilitá trascurabili (negligible)

Approccio asintotico

Usando l'acronimo PPT per *probabilistico di tempo polinomiale* diremo che

Uno schema é sicuro se qualsiasi Adv PPT ha successo nella rottura dello schema con al piú probabilitá trascurabile

Questa nozione é *asintotica* perché la sicurezza dipende dal comportamento dello schema per valori del parametro n sufficientemente grandi.

Esempio 3.2 Disponiamo di uno schema asintoticamente sicuro. Un Adv che

- esegue per n^3 minuti (tempo polinomiale)
- ha successo con prob. $2^{40} \cdot 2^{-n}$ (prob. trascurabile. i.e., $< \frac{1}{p(n)}$)

riesce

Approccio asintotico

- per $n \leq 40$, esegue per 40^3 minuti (6 settimane), ed ha successo con prob. 1.
- per $n = 50$, esegue per 50^3 minuti (3 mesi), ed ha successo con prob. $\approx \frac{1}{1000}$.
- per $n = 500$, esegue per 500^3 minuti (200 anni), ed ha successo con prob. $\frac{1}{2^{-460}}$.

Il parametro di sicurezza n é un meccanismo che permette di *calibrare* la sicurezza dello schema al livello desiderato.

Relativamente agli schemi di cifratura

- guardare al parametro di sicurezza n come alla lunghezza della chiave corrisponde essenzialmente al fatto che il tempo per una ricerca esaustiva cresce esponenzialmente nella lunghezza della chiave

Approccio asintotico

Nota: incrementando n, incrementa anche il tempo di esecuzione per le parti oneste.

Occorre trovare un giusto equilibrio tra

- tempo richiesto alle parti oneste (piú piccolo possibile)
- tempo richiesto all'avversario (piú grande possibile)
- probabilitá di successo (piú piccole possibili)

Esempio 3.3 Computer *piú veloci* giocano a favore delle parti oneste.

Consideriamo un protocollo crittografico in cui:

- le parti oneste eseguono per $10^6 \cdot n^2$ cicli di CPU
- Adv esegue per $10^8 \cdot n^4$ cicli di CPU, ed ha successo con prob. $2^{-\frac{n}{2}}$.

Per computer a 2GHz, ed $n = 80$

- le parti oneste eseguono per $10^6 \cdot 80^2$ cicli di CPU, $3,2$ secondi
- Adv esegue per $10^8 \cdot 80^4$ cicli di CPU, circa 3 settimane, ed ha successo con prob. 2^{-40}.

Per computer a 8GHz, ed $n = 160$ (incrementando la lunghezza della chiave)

- le parti oneste eseguono per $10^6 \cdot 160^2$ cicli di CPU, ancora $3,2$ secondi
- Adv esegue per $10^8 \cdot 160^4$ cicli di CPU, *piú di* 13 *settimane*, ed ha successo *con prob.* 2^{-80}.

Approccio asintotico

Pertanto, computer piú veloci $\Rightarrow$ lavoro di Adv piú difficile.

Nota: anche quando si usa l'approccio asintotico, garanzie di sicurezza concrete sono necessarie in pratica. Come gli esempi mostrano, solitamente una valutazione asintotica puó essere convertita in limitazioni di sicurezza concrete, per ogni valore di n.

Approccio asintotico in dettaglio

Algoritmi efficienti.

A esegue in tempo polinomiale se esiste un polinomio $p(\cdot)$ tale che, per ogni $x \in \{0,1\}^*$, $A(x)$ termina in al piú $p(|x|)$ passi, dove $|x|$ indica la lunghezza dell'input.

Come anticipato, progetteremo schemi in funzione del parametro di sicurezza n

- forniremo, pertanto, il parametro n scritto in unario, i.e., 1^n (una stringa di n valori 1) come input agli algoritmi esplicitamente alcune volte

Le parti possono avere altri input ovviamente

- il tempo di esecuzione degli algoritmi corrispondenti deve essere polinomiale *nella lunghezza totale* dei loro input

Assumiamo per default tutti gli algoritmi come probabilistici: hanno accesso ad un nastro di random bit di lunghezza sufficiente. Dá piú potere agli avversari.

Probabilitá di successo trascurabili.

Definizione 3.4. Una funzione $f : N \to R^+$ é trascurabile (negligible) se, per ogni polinomio positivo p, esiste un n_0 tale che, per tutti gli $n > n_0$, risulta $f(n) < \frac{1}{p(n)}$.

$$\approx$$

Per ogni costante c, esiste un n_0 tale che, per tutti gli $n > n_0$, risulta $f(n) < \frac{1}{n^{-c}}$. Denoteremo una generica funzione trascurabile con *negl*.

Esempio 3.5. Le funzioni

$$2^{-n} \qquad 2^{-\sqrt{n}} \qquad \text{ed} \qquad n^{-\log n}$$

sono tutte funzioni trascurabili.

Approccio asintotico in dettaglio

Tuttavia, esse tendono a zero in modi diversi.

Consideriamo, per esempio, il minimo valore di n per cui ogni funzione é $< 1/n^5$

- $2^{-n} < n^{-5}$ se e solo se $n > 5 \log n$, che vale da $n = 23$
- $2^{-\sqrt{n}} < n^{-5}$ se e solo se $n > 25 \log^2 n$, che vale da $n \approx 3500$
- $n^{-\log n} < n^{-5}$ se e solo se $\log n > 5$, che vale da $n = 33$

Attenzione: sembrerebbe che $2^{-\sqrt{n}}$ tende a zero piú lentamente di $n^{-\log n}$, ma non é cosí. Per tutti gli $n > 65536$ risulta $2^{-\sqrt{n}} < n^{-\log n}$. Tuttavia, per valori di n piú piccoli, una prob. di successo per Adv pari a $n^{-\log n}$ é preferibile ad una pari a $2^{-\sqrt{n}}$.

Lavorare con funzioni trascurabili é tecnicamente vantaggioso perché soddisfano utili proprietá di chiusura.

Approccio asintotico in dettaglio

Proposizione 3.6. Siano $negl_1(n)$ e $negl_2(n)$ funzioni trascurabili

1. la funzione $negl_3(n) = negl_1(n) + negl_2(n)$ é trascurabile
2. per qualsiasi polinomio p, la funzione $negl_4(n) = p(n) \cdot negl_1(n)$

Osservazioni:

- un evento a prob. trascurabile *rimane* trascurabile anche se l'esperimento che potrebbe generarlo viene ripetuto $p(n)$ volte
- d'altra parte, se una funzione g *non* é trascurabile, allora la funzione $f(n) \stackrel{def}{=} \frac{g(n)}{p(n)}$ *non* é trascurabile, per ogni polinomio positivo.

Sicurezza asintotica in sintesi

Ogni definizione di sicurezza consiste di due parti:

- specifica di cosa é una rottura dello schema
- quale potere ha Adv

Il framework generale per le definizioni é il seguente

Uno schema é sicuro se, per ogni Adv PPT A che sferra un attacco di qualche tipo formalmente specificato, e per ogni polinomio positivo p, esiste un intero n_0 tale che, quando $n > n_0$, la probabilitá che A abbia successo é minore di $1/p(n)$.

Nota: nulla é garantito per valori di $n \leq n_0$.

Ragioni alla base della scelta asintotica

Arbitrarie ma conformi alle scelte in teoria della complessitá

Algoritmi PPT:

- usuale per rappresentare computazioni efficienti
- indipendenza dal modello computazionale prescelto
  - Tesi di Church-Turing estesa: tutti i modelli *ragionevoli* sono computazionalmente equivalenti
- proprietá di composizione dei polinomi
  - A invoca subroutines. Se A é polinomiale e le subroutine sono polinomiali $\Rightarrow$ l'intero algoritmo é polinomiale

Probabilitá trascurabili:

- proprietá di chiusura evidenziate in precedenza

Necessitá del rilassamento

Segretezza computazionale vs segretezza perfetta

- rispetto ad Adv efficienti, con piccole probabilitá di successo

Entrambe sono essenziali. Per capirlo, supponiamo di avere uno schema di cifratura in cui $|K| < |M|$

- Dato un cifrato c, Adv puó decifrare c usando tutte le chiavi $k \in K$, ottenendo una lista L di messaggi. $L \not\supseteq M$. Pertanto, rivela informazioni sul messaggio m a cui c corrisponde
- Similmente, se Adv possedesse coppie

$$(m_1, c_1), (m_2, c_2), \ldots, (m_\ell, c_\ell)$$

potrebbe tentare di decifrare le coppie fino a trovare k ed usarla successivamente per decifrare un nuovo c.

Necessitá del rilassamento

Ricerche esaustive come le precedenti permettono ad Adv di aver successo con probabilitá 1 in tempo lineare in $|K|$

$$\Downarrow$$

Dobbiamo escludere tali ricerche, limitando il tempo di esecuzione di Adv

D'altro canto

- se Adv possedesse ancora le coppie

$$(m_1, c_1), (m_2, c_2), \ldots, (m_\ell, c_\ell)$$

potrebbe *scegliere a caso* una chiave e usare le coppie per verificare la correttezza della scelta. Adv esegue in tempo costante e ha successo con probabilitá $1/|K|$

$$\Downarrow$$

Dobbiamo ammettere piccole probabilitá di successo per escludere questi attacchi

Schemi di cifratura computazionalmente sicuri

Definizione 3.7. Uno schema di cifratura a chiave privata é una tripla $\Pi = (Gen, Enc, Dec)$ di algoritmi PPT tale che

1. $k \leftarrow Gen(1^n)$, algoritmo probabilistico di generazione della chiave k
   - dove la chiave $k \in K$ é tale che $|k| \geq n$
2. $c \leftarrow Enc_k(m)$, algoritmo probabilistico di cifratura
   - dove il messaggio $m \in \{0,1\}^*$, la chiave $k \in K$ e il cifrato $c \in \{0,1\}^*$
3. $m := Dec_k(c)$, algoritmo deterministico di decifratura
   - dove il cifrato $c \in \{0,1\}^*$, la chiave $k \in K$ e il messaggio $m \in \{0,1\}^*$
   - $Dec(c)$ restituisce $\perp$ in caso di errore

Correttezza. Per ogni n, per ogni k restituito da $Gen(1^n)$ e per ogni $m \in \{0,1\}^*$, risulta

$$Dec_k(Enc_k(m)) = m$$

Schemi di cifratura computazionalmente sicuri

Note e osservazioni:

- se lo spazio dei messaggi é $\{0,1\}^{\ell(n)}$, allora Π é uno schema di cifratura a chiave privata a lunghezza fissa, per messaggi di lunghezza $\ell(n)$.
- solitamente $Gen(1^n)$ restituisce stringhe di n bit scelte uniformemente a caso
- la definizione é senza stato (occasionalmente considereremo schemi con stato)

Definizione di sicurezza di base:

1. **Modello delle minacce.** Adv é PPT. Osserva un singolo cifrato ottenuto usando una certa chiave. Puó applicare qualsiasi strategia d'attacco.
2. **Garanzie di sicurezza.** Adv non deve essere in grado di acquisire *alcuna informazione* circa il messaggio in chiaro m a partire dal cifrato c.

Sicurezza semantica ed indistinguibilitá

La nozione di *sicurezza semantica* formalizza ció.

$\Downarrow$

É difficile da maneggiare

$\Downarrow$

Esiste una definizione *equivalente* piú semplice

$\Downarrow$

É la nozione di *Indistinguibilitá*

Indistinguibilitá

Nel contesto della segretezza perfetta abbiamo considerato l'esperimento $PrivK^{eav}_{A,\Pi}$

$PrivK^{eav}_{A,\Pi}$

1. $m_0, m_1 \leftarrow A$ (sceglie due messaggi)
2. il challenger calcola $c \leftarrow Enc_k(m_b)$, dove $b \leftarrow \{0,1\}$ e $k \leftarrow Gen(1^n)$
3. A riceve c e dá in output $b' \in \{0,1\}$
4. Se $b = b'$, l'output dell'esperimento é 1 (A vince); altrimenti, 0.

Lo schema Π é sicuro se A vince con probabilitá $1/2$, i.e., non c'é strategia migliore per indovinare che *scegliendo a caso*

Indistinguibilitá

Nel caso computazionale:

- A é PPT
- A puó vincere con probabilitá *trascurabilmente* migliore di $1/2$
- L'esperimento dipende da n, il parametro di sicurezza

$PrivK^{eav}_{A,\Pi}(n)$

1. A ottiene 1^n e dá in output $m_0, m_1 \leftarrow A$ tali che $|m_0| = |m_1|$
2. il challenger calcola $c \leftarrow Enc_k(m_b)$, dove $b \leftarrow \{0,1\}$ e $k \leftarrow Gen(1^n)$
3. $A(1^n)$ riceve c e dá in output $b' \in \{0,1\}$
4. Se $b = b'$, l'output dell'esperimento é 1 ($A(1^n)$ vince); altrimenti, 0.

Indistinguibilitá

Definizione 3.8. Uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ ha *cifrature indistinguibili* in presenza di un avversario che ascolta (eavesdropper) o é EAV-sicuro se, per ogni Adv A PPT, esiste una funzione trascurabile *negl* tale che, per tutti gli n si ha:

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

dove la probabilitá é calcolata su

- randomness usata da A
- randomness usata nell'esperimento
  - scelta della chiave
  - scelta del bit b
  - random bit usati da $Enc_k(\cdot)$

Indistinguibilitá

Nota: qualsiasi schema di cifratura *perfettamente segreto* ha cifrature indistinguibili in presenza di un eavesdropper.

Faremo vedere che esistono schemi con "chiavi piú corte"

Esiste una formulazione equivalente: l'idea di fondo é che *ogni Adv PPT si comporta allo stesso modo sia che veda una cifratura di m_0 che di m_1*

Definendo $PrivK^{eav}_{A,\Pi}(n, b)$ con $b \in \{0, 1\}$ e l'output di A con $out_A(PrivK^{eav}_{A,\Pi}(n, b))$, diamo la seguente

Definizione 3.9. $\Pi = (Gen, Enc, Dec)$ é EAV-sicuro se, per ogni Adv A PPT, esiste una funzione trascurabile *negl* tale che, per tutti gli n si ha:

$$Pr[out_A(PrivK^{eav}_{A,\Pi}(n, 0)) = 1] - Pr[out_A(PrivK^{eav}_{A,\Pi}(n, 1)) = 1] \leq negl(n).$$

Indistinguibilitá

Nota: nella definizione *non* richiediamo ad uno schema di nascondere la lunghezza del messaggio da cifrare. Nelle informazioni in cui questa informazione é importante occorre porre rimedio (e.g., estendendo i messaggi ad una lunghezza fissa)

Sicurezza Semantica

Definizione 3.12. Uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ é semanticamente sicuro in presenza di un eavesdropper se, per ogni Adv A PPT, esiste un Adv A' PPT tale che, per qualsiasi $Samp(1^n)$ PPT e per ogni coppia di funzioni f ed h, calcolabili in tempo polinomiale, esiste una funzione trascurabile $negl$ per cui si ha:

$$|Pr[A(1^n, Enc_k(m), h(m)) = f(m)] - Pr[A'(1^n, |m|, h(m)) = f(m)]| \leq negl(n),$$

dove la prima probabilitá é calcolata su

- scelta uniforme di $k \in \{0,1\}^n$
- random bit usati da $Samp(1^n)$
- random bit usati da A
- random bit usati da $Enc_k(\cdot)$

e la seconda su

- random bit usati da $Samp(1^n)$ e random bit usati da A'

Sicurezza Semantica

Teorema 3.13. $\Pi = (Enc, Dec)$ ha cifrature indistinguibili in presenza di un eavesdropper *se e solo se* é semanticamente sicuro in presenza di un eavesdropper.

$$\Downarrow$$

Possiamo usare la definizione piú semplice di indistinguibilitá come definizione di lavoro!

Costruzioni di schemi di cifratura sicuri

Abbiamo bisogno di due tipi di blocchi di base:

- Generatori pseudocasuali (pseudorandom generator, PRG)
- Cifrari a flusso (stream cipher)

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

Studiati a partire dagli anni '40

- progettati per superare test statistici ad hoc: paritá, primo bit, ...
- 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 uniforme

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

- pseudocasualitá: proprietá di una distribuzione di stringhe
- uniformitá: proprietá di una distribuzione di stringhe

Distribuzione pseudocasuale

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

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

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

Risulta, cioé, 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 é tanto buona quanto una uniforme.*

Generatore pseudocasuale (informale)

Sia $G : \{0,1\}^n \leftarrow \{0,1\}^\ell$ 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 é PRG se e solo se *Dist* é 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 $s \in \{0,1\}^n$, $G(s)$ é una stringa di $\ell(n)$ bit. Diremo che G é un PRG se valgono le seguenti condizioni:

1. **Espansione**: per ogni n risulta $\ell(n) > n$
2. **Pseudocasualitá**: 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 probabilitá é 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 $G : \{0,1\}^n \leftarrow \{0,1\}^{n+1}$ tale che $G(s) = s || \bigoplus_{i=1}^{n} s_i$.

Distinguisher D:
su input ω dá in output 1 se e solo se il bit finale di ω é l' *xor* di tutti i precedenti.

Risulta:

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

Poiché la differenza é

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

costante (non trascurabile), G non é pseudocasuale.

Osservazioni

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

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

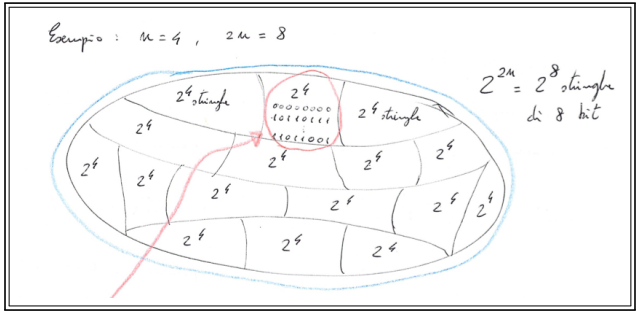
Poiché ci sono 2^n possibili semi, G puó generare *al piú* 2^n delle 2^{2n} possibili stringhe.

Equivalentemente, la frazione di stringhe di $2n$ bit che G puó generare é

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

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

Osservazioni



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

Osservazioni

Data una quantitá illimitata di tempo é pertanto banale distinguere.

Un distinguisher D di **tempo esponenziale** tenta tutti i semi $s \in \{0,1\}^n$ e dá 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 $Pr[G(s) = \omega] = 1/2^n$. 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 $s \in \{0,1\}^n$.

Ció **non** contraddistingue la pseudocasualitá di G perché D **non** é 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.

Prove (dimostrazioni) per riduzione

Provare che una costruzione é computazionalmente sicura significa

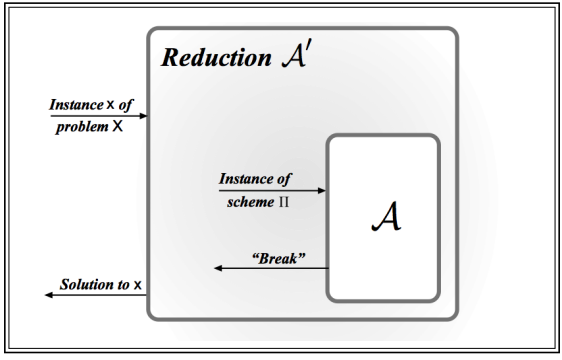
- far affidamento su assunzioni (non provate)
  - un problema matematico é difficile
  - qualche primitiva crittografica é sicura
- provare che, basandosi sulle assunzioni fatte, la costruzione risulta sicura.

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 é difficile, i.e., non puó essere risolto con algoritmi PPT.

Vogliamo provare che la costruzione Π é sicura

Struttura di una riduzione

1. Fissiamo un Adv A PPT che attacca Π con probabilitá 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:

   2.1 A pensa di star interagendo con Π, i.e., ha la *stessa vista* (o molto simile) che ha quando interagisce realmente con Π

   2.2 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 probabilitá almeno $1/p(n)$

Struttura di una riduzione

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

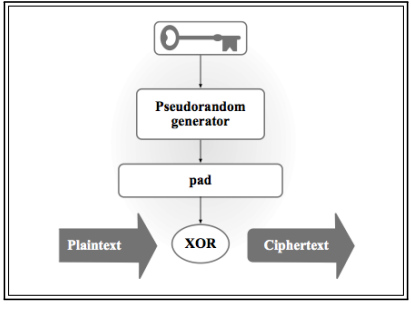
4. Data l'assunzione di difficoltá sul problema X, concludiamo che *nessun* Adv A PPT puó avere successo nel rompere Π con probabilitá non trascurabile. Quindi, Π é computazionalmente sicuro.

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 é 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 probabilitá 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 abilitá nel capire quale messaggio tra m_0 ed m_1 é stato cifrato

Probabilitá di successo di D $\Rightarrow$ relata alla probabilitá 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. dá c ad A ed ottiene da A il bit b'
4. se $b' = b$, dá in output 1; altrimenti, dá 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 é PPT se A é un Adv PPT.

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

É esattamente il one-time pad: $\widetilde{Gen}\,(1^n)$ riceve in input il parametro di sicurezza n e dá 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 é una stringa uniforme in $\{0,1\}^{\ell(n)}$ allora
   - la vista di A quando eseguito come subroutine da D = la vista di A in $PrivK^{eav}_{A,\widetilde{\Pi}}(n)$
   - poiché D dá 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 é una stringa uniforme in $\{0,1\}^n$ allora
   - la vista di A quando eseguito come subroutine da D = la vista di A in $PrivK^{eav}_{A,\Pi}(n)$
   - poiché D dá 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

Ma G per ipotesi é un PRG e D é 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

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

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

Cifratura sicura con un PRG

Commenti:

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

Attacchi di tipo Chosen-Plaintext

Adv: ha l'abilitá di esercitare *controllo parziale* su ció che una parte onesta cifra

1. puó scegliere $m_1, m_2, \ldots, m_t$ e forzare Alice a cifrarli ed inviarli
2. puó osservare i cifrati corrispondenti $c_1, c_2, \ldots, c_t$ inviati da Alice sul canale
3. osserva un nuovo cifrato c, prodotto *autonomamente* da Alice, che diventa il target dell'attacco

Anche in questo caso, definiremo la sicurezza di uno schema in accordo alla nozione di indistinguibilitá

- al passo 3. il cifrato c corrisponde alla cifratura di uno tra $\{m_0, m_1\}$, *noti* ad Adv, e richiederemo l'incapacitá di Adv di capire *a quale dei due* effettivamente corrisponde

Attacchi di tipo Chosen-Plaintext

Nota: gli attacchi di tipo *known-plaintext* sono un caso particolare. Adv conosce ma *non sceglie* $m_1, \ldots, m_t$.

Sono una preoccupazione realistica? Sí!

Nel libro di testo vengono riportati due esempi storici:

- Tedeschi: seconda guerra mondiale
- Us Navy, battaglia di Midway, 1942

Leggeteli!

Come facciamo a modellare la capacitá di Adv di disporre di cifrati corrispondenti a messaggi di propria scelta?

Esperimenti

Abbiamo giá usato esperimenti per definire la nozione di indistinguibilitá perfetta e computazionale

Molte definizioni in crittografia vengono fornite utilizzando degli esperimenti, in cui un Challenger sfida un Adv affinché abbia successo in un determinato task

Gli esperimenti permettono di *astrarre* e *modellare* scenari reali in modo semplice

Per modellare lo scenario di un attacco chosen-plaintext abbiamo bisogno di uno stratagemma per fornire ad *Adv* i cifrati corrispondenti ai messaggi che sceglie.

Useremo un **oracolo** O: scatola nera che cifra messaggi usando una chiave k

- Adv non conosce k
- Adv invia richieste di cifratura, dette *query*, ad O specificando m ed ottenendo in risposta $Enc_k(m)$.
- se $Enc_k()$ é randomizzato, O usa random bit *nuovi* ogni volta che riceve una query
- puó inviare, adattivamente, quante query vuole

Esperimenti

Siano $\Pi = (Gen, Enc, Dec)$, Adv A, n parametro di sicurezza.
Indichiamo con $A^{O(\cdot)}$ un Adv (algoritmo) che ha accesso all'oracolo $O(\cdot)$.

$PrivK^{cpa}_{A,\Pi}(n)$ (Gestito da un Challenger)

1. Genera $k \leftarrow Gen(1^n)$ e *setta* l'oracolo $O(\cdot)$
2. $A^{O(\cdot)}(1^n)$ dá in output m_0 ed m_1 tali che $|m_0| = |m_1|$
3. Sceglie $b \leftarrow \{0,1\}$ e calcola $c \leftarrow Enc_k(m_b)$
4. $A^{O(\cdot)}(1^n)$ riceve c e dá in output $b' \in \{0,1\}$
5. Se $b = b'$, l'output dell'esperimento é 1 ($A^{O(\cdot)}(1^n)$ vince); altrimenti, 0.

Sicurezza rispetto ad attacchi chosen-plaintext

Definizione 3.22. Uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ ha *cifrature indistinguibili* rispetto ad attacchi di tipo chosen plaintext (CPA-sicuro) se, per ogni Adv A PPT, esiste una funzione trascurabile *negl* tale che:

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

dove la probabilitá é calcolata su

- randomness usata da A
- randomness usata nell'esperimento

Sicurezza CPA per cifrature multiple

Per la formalizzazione, usiamo un approccio diverso dal precedente.

Un Adv ha accesso ad un oracolo *left-or-right*, denotato con $LR_{k,b}$, che, su input m_0 ed m_1, restituisce $c \leftarrow Enc_k(m_b)$

$PrivK^{LR-cpa}_{A,\Pi}(n)$ (Gestito da un Challenger)

1. Genera $k \leftarrow Gen(1^n)$
2. Sceglie $b \leftarrow \{0,1\}$
3. $A^{LR_{k,b}(\cdot)}(1^n)$ dá in output $b' \in \{0,1\}$
4. Se $b = b'$, l'output dell'esperimento é 1 ($A^{LR_{k,b}(\cdot)}(1^n)$ vince); altrimenti, 0.

Differenza con l'approccio precedente: le coppie $(m_{0,i}, m_{1,i})$ sono scelte *adattivamente* invece che in un sol colpo.

Sicurezza per messaggi multipli rispetto ad attacchi chosen-plaintext

Definizione 3.23. Uno schema di cifratura a chiave privata $\Pi = (Gen, Enc, Dec)$ ha *cifrature multiple indistinguibili* rispetto ad attacchi di tipo chosen plaintext se, per ogni Adv A PPT, esiste una funzione trascurabile *negl* tale che:

$$Pr[PrivK_{A,\Pi}^{LR-cpa}(n) = 1] \leq \frac{1}{2} + negl(n),$$

dove la probabilitá é calcolata su

- randomness usata da A
- randomness usata nell'esperimento

Ovviamente,

$$\Pi \text{ CPA-sicuro per cifrature multiple} \Rightarrow \Pi \text{ CPA-sicuro}$$

Vale anche l'inverso!

Conseguenze

Teorema 3.4. Ogni schema di cifratura a chiave privata CPA-sicuro risulta *anche* CPA-sicuro per cifrature multiple.

Discende che:

- é sufficiente provare che uno schema é CPA-sicuro (per una sola cifratura) per ottenere *gratuitamente* che é CPA-sicuro per cifrature multiple
- permette di concentrarci su schemi di cifratura per messaggi di lunghezza fissata

$$\Pi = (Gen, Enc, Dec) \text{ CPA-sicuro per messaggi di 1 bit}$$

$$\Downarrow$$

$$\Pi' = (Gen', Enc', Dec') \text{ CPA-sicuro per messaggi di lunghezza arbitraria}$$

Conseguenze

Struttura dello schema per messaggi di lunghezza arbitraria

- $Gen' = Gen$
- $Enc'_k(m) = Enc_k(m_1) \dots Enc_k(m_\ell)$, dove $m = m_1 \dots m_\ell$, ed $m_i \in \{0,1\}$
- $Dec'_k(c) = Dec_k(c_1) \dots Dec_k(c_\ell)$, dove $c = c_1 \dots c_\ell$, e $c_i \in C$ per ogni i

Osservazione:

La cifratura puó essere vista come la cifratura di messaggi multipli. Pertanto, la sicurezza CPA di Π' discende dalla sicurezza CPA di Π per messaggi multipli.

Costruzione di schemi CPA-sicuri

Abbiamo bisogno di uno strumento nuovo

- le funzioni pseudocasuali (pseudorandom function, o PRF in breve)

Generalizzano la nozione di generatore pseudocasuale

- i PRG producono *stringhe che sembrano casuali*
- le PRF sono *funzioni che sembrano casuali*

Non ha senso parlare di *una* funzione fissata

Dobbiamo considerare una *distribuzione di funzioni*.

Funzioni *parametrizzate da una chiave* (keyed function) inducono naturalmente una distribuzione di funzioni

Funzioni pseudocasuali

Una funzione parametrizzata da una chiave

$$F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

é una funzione con due input, in cui il primo é chiamato chiave e viene denotato con k.

F é efficiente se $\exists$ un algoritmo di tempo polinomiale per calcolare $F(k,x)$, dati k e x

In usi tipici, k viene scelto e fissato. Per cui

$$F_k(x) = F(k,x) \quad \text{ovvero} \quad F_k : \{0,1\}^* \to \{0,1\}^* \quad \text{(funzione di una variabile)}$$

Nella nostra trattazione, il parametro di sicurezza n parametrizza tre funzioni:

- $\ell_{key}(n)$ lunghezza della chiave
- $\ell_{in}(n)$ lunghezza dell'input
- $\ell_{out}(n)$ lunghezza dell'output

Funzioni pseudocasuali

Per ogni $k \in \{0,1\}^{\ell_{key}(n)}$, F_k é definita solo per $x \in \{0,1\}^{\ell_{in}(n)}$ ed ha output $y \in \{0,1\}^{\ell_{out}(n)}$.

Siano $\ell_{key}(n) = \ell_{in}(n) = \ell_{out}(n) = n$ (F *preserva la lunghezza*).

Una funzione $F(\cdot,\cdot)$ parametrizzata da una chiave induce una distribuzione di funzioni

- scegliendo una chiave uniforme $k \in \{\}^n$
- considerando la funzione di una singola variabile F_k risultante

> F é pseudocasuale se F_k, per k scelta uniformemente a caso, é *indistinguibile* da una funzione scelta uniformemente a caso dall'insieme di tutte le funzioni aventi lo stesso dominio e lo stesso codominio

Funzioni pseudocasuali

Per formalizzare la nozione, abbiamo bisogno di chiarire alcuni aspetti.

Per esempio, cosa significa *scegliere una funzione a caso*?

$$\text{Sia } Func_n = \{ \text{ tutte le funzioni } f : \{0,1\}^n \to \{0,1\}^n \}$$

Una funzione puó essere rappresentata con una tabella con 2^n righe, ciascuna di n bit.

$f \in Func_n \to$

1	$f(1)$
2	$f(2)$
...	...
i	$f(i)$
...	...
2^n	$f(2^n)$

$\leftarrow$ valore della funzione sull'i-esima stringa di input

Funzioni pseudocasuali

Se concatenassimo tutte le righe della tabella, potremmo vedere f come una stringa di $2^n \cdot n$ bit, cioé:

$f(1)$	$\ldots$	$f(i)$	$\ldots$	$f(2^n)$

Pertanto,

$$|Func_n| = 2^{2^n \cdot n} \quad \leftarrow \text{ numero di funzioni di } n\text{-bit in } n\text{-bit}$$

Si noti che lo stesso insieme di funzioni $Func_n$ puó essere visto come una tabella.

scegliere una funzione a caso
$\approx$
scegliere una riga della tabella a caso

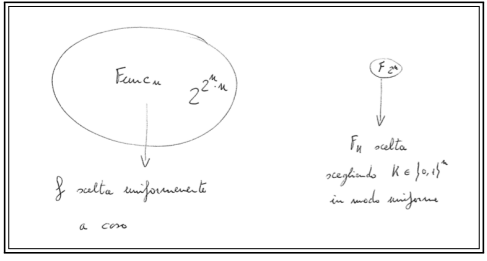
1	$f_1(\cdot)$
$\ldots$	$\ldots$
i	$f_i(\cdot)$
$\ldots$	$\ldots$
$2^{2^n \cdot n}$	$f_{2^{2^n \cdot n}}(\cdot)$

Funzioni pseudocasuali

D'altra parte, quest'ultima operazione é equivalente a vedere la riga della tabella come *una riga di elementi scelti al volo*, ogni volta che f viene valutata su un nuovo input.

Essenzialmente la riga viene riempita *volta per volta*.

Viceversa, F_k, per k uniforme, viene scelta su un insieme $\mathcal{F}$ di al piú 2^n funzioni.



Funzioni pseudocasuali

Dire che F é pseudocasuale significa che, *nonostante* la notevole differenza evidenziata, il *comportamento* di f e di F_k sembra lo stesso a qualsiasi algoritmo PPT che cerca di distinguere tra i due casi.

Come formalizzare *distinguere*?

Prima idea: dare all'algoritmo D che distingue (PPT) le descrizioni di F_k ed f.

D dovrebbe dare in output 1 all'incirca con la stessa probabilitá nei due casi.

Ma ... f ha *lunghezza esponenziale* ($2^n \cdot n$ bit) e D, che é PPT, *non* puó neanche leggere la sua descrizione!

Seconda idea: dare a D accesso ad un *oracolo* $O(\cdot)$ che o implementa F_k, per k uniforme, o implementa f, per f uniforme.

- D puó chiedere il valore della funzione su un numero polinomiale di input x
- non chiede mai due volte il valore per lo stesso x
- al termine deve decidere se ha interagito con F_k o f

Definizione

Definizione 3.25. Sia $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ una funzione con chiave efficiente che preserva la lunghezza. F é una funzione pseudocasuale se, per ogni distinguisher D PPT, esiste una funzione trascurabile *negl* tale che:

$$|Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1]| \leq negl(n),$$

dove la prima probabilitá é calcolata su

- scelta uniforme di k
- random bit di D

e la seconda su

- scelta uniforme di f
- random bit di D

Pseudocasualitá

Nota: Ovviamente D *non* riceve la chiave k!

Altrimenti risulterebbe banale per D distinguere.

Infatti, chiedendo all'oracolo $O(\cdot)$ una valutazione su x e ricevendo $O(x)$, potrebbe calcolare $F_k(x)$ e controllare che $F_k(x) = O(x)$. Se l'uguaglianza sussiste, D con altissima probabilitá sta interagendo con F_k. Piú valutazioni corroborerebbero l'ipotesi.

$$\Downarrow$$

Se k diventa noto, la pseudocasualitá non c'é piú!

Esempio di funzione non pseudocasuale

Esempio 3.6. Sia F una funzione con chiave che preserva la lunghezza, definita da

$$F(k, x) = k \oplus x.$$

Per ogni input x, il valore $F_k(x)$ é uniformemente distribuito quando k viene scelto in modo uniforme.

F non é pseudocasuale poiché i suoi valori su ogni coppia di punti sono correlati

Infatti, D:

- chiede all'oracolo valutazioni su x_1 e x_2
- ottiene $y_1 = O(x_1)$ e $y_2 = O(x_2)$
- se $y_1 \oplus y_2 = x_1 \oplus x_2$, dá in output 1; altrimenti, dá 0.

Esempio di funzione non pseudocasuale

É facile vedere che:

- se $O \equiv F_k$, per ogni k, D dá in output 1 con probabilitá 1, poiché

$$y_1 \oplus y_2 = (x_1 \oplus k) \oplus (x_2 \oplus k) = x_1 \oplus x_2$$

- se $O \equiv f$, D dá in output 1 con probabilitá

$$Pr[y_1 \oplus y_2 = x_1 \oplus x_2] = Pr[y_2 = x_1 \oplus x_2 \oplus y_1] = 2^{-n}$$

La differenza $|1 - 1/2^n|$ ovviamente non é trascurabile

$$\Downarrow$$

F *non* é pseudocasuale!

Permutazioni pseudocasuali

Sia $Perm_n$ l'insieme di tutte le permutazioni su $\{0,1\}^n$.

Nota che

- $f \in Perm_n$ puó essere vista ancora come una riga in una tabella
- ogni due righe della tabella sono differenti
- $|Perm_n| = 2^n!$

F é una permutazione con chiave se $\ell_{in}(n) = \ell_{out}(n)$ e per ogni $k \in \{0,1\}^{\ell_{key}(n)}$ la funzione

$$F_k : \{0,1\}^{\ell_{in}(n)} \to \{0,1\}^{\ell_{out}(n)}$$

é uno a uno.

Il valore $\ell_{in}(n)$ si dice anche *lunghezza del blocco* di F.

Considereremo il caso in cui $\ell_{key}(n) = \ell_{in}(n) = \ell_{out}(n) = n$.

Permutazioni pseudocasuali

F é efficiente se esiste un algoritmo di tempo polinomiale per calcolare $F(k, x)$, dati k ed x, cosí come un algoritmo di tempo polinomiale per calcolare $F_k^{-1}(y)$, dati k e y.

$$\Updownarrow$$

F efficientemente calcolabile ed invertibile, data k.

La pseudocasualitá é definita esattamente come per le funzioni.

Nota: quando la lunghezza del blocco é sufficientemente lunga, una permutazione casuale é indistinguibile da una funzione casuale.

$$\Downarrow$$

Permutazioni pseudocasuali

$$\text{Funzione uniforme} \overset{\text{"appare identica"}}{\approx} \text{Permutazione uniforme}$$

... a meno che il distinguisher D non trovi x ed y tali che $f(x) = f(y)$.

La probabilitá di un tale evento é trascurabile utilizzando un numero polinomiale di query.

Proposizione 3.27. Se F é una permutazione pseudocasuale e $\ell_{in}(n) \geq n$, allora F é anche una funzione casuale.

Alcune costruzioni crittografiche richiedono alle parti oneste di usare anche F_k^{-1}. Pertanto, Adv puó conoscere anche questi valori

Abbiamo bisogno di una nozione forte, che tenga in conto anche questa possibilitá dell'Adv, che possiamo modellare con un *accesso ad un oracolo* per F_k^{-1}.

Permutazioni pseudocasuali forti (strong pseudorandom permutation)

Definizione 3.28. Sia $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ una permutazione con chiave efficiente che preserva la lunghezza. F é una permutazione pseudocasuale forte se, per ogni distinguisher D PPT, esiste una funzione trascurabile *negl* tale che:

$$|Pr[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1]| \leq negl(n),$$

dove la prima probabilitá é calcolata su

- scelta uniforme di k
- random bit di D

e la seconda su

- scelta uniforme di f
- random bit di D

Nota: una permutazione pseudocasuale forte é *anche* una permutazione pseudocasuale.

Nella pratica i *cifrari a blocchi* vengono progettati per essere istanziazioni sicure di permutazioni pseudocasuali forti, con una lunghezza della chiave e del blocco fissate.

Funzioni pseudocasuali e generatori pseudocasuali

Un generatore pseudocasuale G da una funzione pseudocasuale F si costruisce facilmente:

$$G(s) = F_s(1)||F_s(2)||\ldots||F_s(\ell)$$

per ogni valore di ℓ desiderato.

Idea della prova: se sostituiamo F_s con $f \in Func_n$

$$G'(s) = f(1)||f(2)||\ldots||f(\ell) \qquad \text{(uniforme)}$$

$$\Downarrow$$

$$G(s) = F_s(1)||F_s(2)||\ldots||F_s(\ell) \qquad \text{(pseudocasuale)},$$

perché, se non lo fosse, esisterebbe un D PPT che distingue F_s da f.

Piú in generale, possiamo costruire uno stream cipher a partire da F.

Generatori pseudocasuali e funzioni pseudocasuali

Un generatore pseudocasuale G dá immediatamente una funzione pseudocasuale F *con lunghezza di blocco piccola*. Sia

$$G : \{0,1\}^n \to \{0,1\}^{2^{t(n)} \cdot n}$$

un generatore con fattore di espansione $2^{t(n)} \cdot n$.

Per calcolare $F_k(i)$:

- calcoliamo $G(k)$
- rappresentiamo l'output con una tabella
- prendiamo la i-esima riga tra le $2^{t(n)}$ di n bit

r_1
...
r_i
...
$r_{2^{t(n)}}$

Questa costruzione é efficiente *solo se* $t(n) = O(\log n)$.

Infatti

$$2^{t(n)} \cdot n = 2^{c \log n} \cdot n = n^c \cdot n = poly(n)$$

Nota che

$$F : \{0,1\}^n \times \{0,1\}^{c \log n} \to \{0,1\}^n$$

associa stringhe di n bit a stringhe di input di $c \log n$ bit.

Una costruzione generale esiste ma é piú complicata. Ci torneremo nel seguito.

Cifratura CPA sicura da funzioni pseudocasuali

Costruiremo uno schema di cifratura per messaggi di lunghezza fissa

Ne possiamo ottenere facilmente uno per lunghezze arbitrarie applicando i risultati precedenti

Primo tentativo:

$$Enc_k(m) = F_k(m)$$

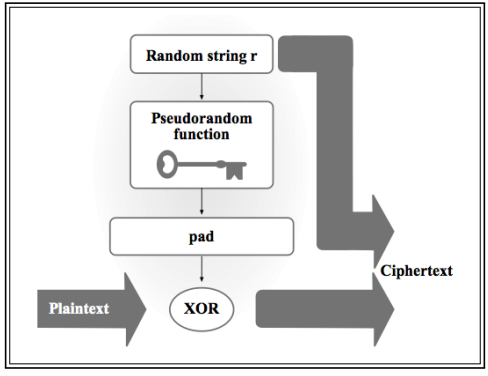
Deterministico ... non funziona ... (stesso messaggio $\Rightarrow$ stesso cifrato)

Approccio giusto:

Applichiamo F_k ad una stringa casuale r per produrre un "pad" pseudocasuale.

Cifriamo calcolando l'xor tra il pad ed il messaggio m.

Idea


Random string r
Pseudorandom function
pad
Plaintext
XOR
Ciphertext

Specifica dello schema

CONSTRUCTION 3.30

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen: on input 1^n, choose uniform $k \in \{0,1\}^n$ and output it.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose uniform $r \in \{0,1\}^n$ and output the ciphertext

$$c := \langle r, F_k(r) \oplus m \rangle.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

$$m := F_k(r) \oplus s.$$

Teorema 3.31. Se F é pseudocasuale, allora la Costruzione 3.30 realizza uno schema di cifratura CPA-sicuro per messaggi di lunghezza n.

Dim. Nota preliminare: le prove di sicurezza per schemi basati su PRF procedono solitamente in due fasi

- Prima fase: consideriamo una versione "ipotetica" della costruzione in cui la funzione pseudocasuale viene sostituita da una funzione casuale. Mostriamo che questa modifica *non* influisce sulla probabilitá di successo di Adv
- Seconda fase: analizziamo lo schema ipotetico che utilizza la funzione casuale

Dimostrazione

Sia $\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$ costruito a partire da $\Pi = (Gen, Enc, Dec)$, tale che

- $\widetilde{\Pi}$ usa $f \in Func_n$ scelta uniformemente a caso
- Π usa F_k, dove k é scelta uniformemente a caso

Ovviamente, $\widetilde{\Pi}$ *non* é efficiente: f richiede spazio esponenziale in n per la memorizzazione.

Per ogni Adv A PPT sia $q(n)$ un limite superiore al numero di query che $A(1^n)$ rivolge al suo oracolo per la cifratura ($q(n)$ deve essere un polinomio)

Mostriamo che esiste una funzione trascurabile *negl* tale che

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

Dimostrazione

Procediamo per riduzione: usiamo A per costruire un distinguisher D per la funzione pseudocasuale F

$$\text{Se } A \text{ ha successo} \Rightarrow D \text{ distingue}$$

Precisamente:

- D ha accesso all'oracolo $O(\cdot)$ ed il suo scopo é stabilire se la funzione é "F_k, per k uniforme in $\{0,1\}^n$" oppure "$f \in Func_n$, uniforme".
- D *emula* l'esperimento $PrivK^{cpa}_{A,?}(n)$ per A ed osserva se A ha successo.

Distinguisher

Distinguisher D:
D is given input 1^n and access to an oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$.

1. Run $\mathcal{A}(1^n)$. Whenever $\mathcal{A}$ queries its encryption oracle on a message $m \in \{0,1\}^n$, answer this query in the following way:
   (a) Choose uniform $r \in \{0,1\}^n$.
   (b) Query $\mathcal{O}(r)$ and obtain response y.
   (c) Return the ciphertext $\langle r, y \oplus m \rangle$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs messages $m_0, m_1 \in \{0,1\}^n$, choose a uniform bit $b \in \{0,1\}$ and then:
   (a) Choose uniform $r \in \{0,1\}^n$.
   (b) Query $\mathcal{O}(r)$ and obtain response y.
   (c) Return the challenge ciphertext $\langle r, y \oplus m_b \rangle$ to $\mathcal{A}$.
3. Continue answering encryption-oracle queries of $\mathcal{A}$ as before until $\mathcal{A}$ outputs a bit b'. Output 1 if $b' = b$, and 0 otherwise.

Analisi

D computa in tempo polinomiale poiché A computa in tempo polinomiale. Inoltre, si noti che:

1. Se l'oracolo di D contiene al suo interno una funzione **pseudocasuale**

$$\text{Vista di } A \text{ come subroutine di } D = \text{ Vista di } A \text{ in } PrivK^{cpa}_{A,\Pi}(n)$$

2. Se l'oracolo di D contiene al suo interno una funzione **casuale**

$$\text{Vista di } A \text{ come subroutine di } D = \text{ Vista di } A \text{ in } PrivK^{cpa}_{A,\widetilde{\Pi}}(n)$$

Pertanto, possiamo dire che:

Analisi

1. $\Rightarrow Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] = Pr[PrivK^{cpa}_{A,\Pi}(n) = 1]$
2. $\Rightarrow Pr_{f \leftarrow Func_n}[D^{f(\cdot)}(1^n) = 1] = Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1]$

Ma l'assunzione che F é pseudocasuale implica che $\exists$ $negl(n)$ tale che:

$$|Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] - Pr_{f \leftarrow Func_n}[D^{f(\cdot)}(1^n) = 1]| \leq negl(n)$$

$$\Updownarrow$$

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

Pertanto possiamo analizzare lo schema ipotetico.

Successo di A nello schema ipotetico

Mostriamo che

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

Nota che, ogni volta che un messaggio m viene cifrato, in $PrivK^{cpa}_{A,\widetilde{\Pi}}(n)$ viene scelto un $r \in \{0,1\}^n$ uniforme, ed il cifrato risulta

$$< r, f(r) \oplus m >$$

Sia r^* la stringa usata per produrre il cifrato di sfida, cioé

$$c^* :=< r^*, f(r^*) \oplus m_b >$$

Possono verificarsi due casi:

Successo di A nello schema ipotetico

- Il valore di r^* *non* é mai usato prima da $O(\cdot)$ per rispondere alle query di A

  $\Rightarrow$ A non sa nulla circa $f(r^*)$, che risulta uniforme ed indipendentemente distribuito dal resto dell'esperimento

  $\Rightarrow Pr[PrivK^{cpa}_{A,\tilde{\Pi}}(n) = 1] = Pr[b' = b] = 1/2$

- Il valore di r^* é stato usato in precedenza

  $\Rightarrow$ A puó capire facilmente se é stato cifrato m_0 o m_1. Infatti, disponendo di $f(r^*)$, poiché $c^* :=< r^*, f(r^*) \oplus m_b >$, risulta

  $$f(r^*) \oplus (f(r^*) \oplus m_b) = m_b.$$

Il valore $f(r^*)$ puó essere recuperato dalla query in cui r^* é usato: se A ha ricevuto dall'oracolo, per qualche m, il cifrato $c :=< r^*, s >=< r^*, f(r^*) \oplus m >$, allora

$$s \oplus m = f(r^*).$$

Successo di A nello schema ipotetico

Tuttavia, poiché A effettua al piú $q(n)$ query all'oracolo, al piú $q(n)$ valori distinti di r vengono usati, scelti indip. e uniformemente in $\{0,1\}^n$.

Pertanto, la prob. che r^*, scelto uniformemente, sia uguale ad un r precedente é al piú $q(n)/2^n$.

Indichiamo con **Repeat** l'evento che r^* sia uguale a qualche r scelto prima.

Risulta:

$$\begin{aligned} Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n)=1] &= Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n)=1 \wedge \textbf{ Repeat}] + Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n)=1 \wedge \overline{\textbf{ Repeat}}] \\ &\leq Pr[\textbf{ Repeat}] + Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n)=1 | \overline{\textbf{ Repeat}}] \cdot Pr[\overline{\textbf{ Repeat}}] \\ &\leq Pr[\textbf{ Repeat}] + Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n)=1 | \overline{\textbf{ Repeat}}] \\ &\leq q(n)/2^n + 1/2. \end{aligned}$$

Successo di A nello schema ipotetico

Poiché abbiamo mostrato che

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

si ha:

$$\begin{aligned} Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] &\leq Pr[PrivK^{cpa}_{A,\widetilde{\Pi}}(n) = 1] + negl(n) \\ &\leq q(n)/2^n + 1/2 + negl(n) \\ &\leq 1/2 + q(n)/2^n + negl(n) \\ &\leq 1/2 + negl'(n). \end{aligned}$$

Pertanto, Π é CPA-sicuro.

Modalitá operative

Esistono diversi modi per cifrare in maniera sicura messaggi lunghi usando

- Stream cipher
- Cifrari a blocchi

Modalitá operative per stream cipher

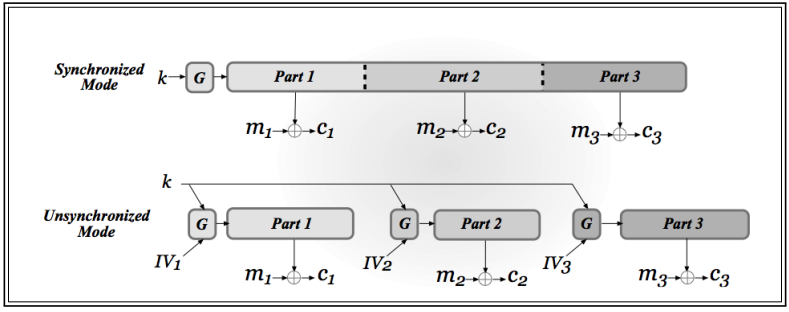
Abbiamo visto con la costruzione 3.17 (costruzione con PRG) come cifrare messaggi di lunghezza fissata in modo EAV-sicuro.

Gli stream cipher possono essere visti come PRG *flessibili*.

Possono essere utilizzati in

- modo sincrono (con stato)
- modo asincrono (senza stato)

Cifratura con stream cipher: sincrona e asincrona



Sincrono: viene generata una lunga stringa pseudocasuale e una parte diversa viene usata per ogni messaggio

Modalitá operative per cifrari a blocchi

Abbiamo visto uno schema di cifratura CPA-sicuro basato su PRF (cifrario a blocchi)

- la lunghezza del cifrato é doppia rispetto alla lunghezza del messaggio

Le modalitá operative per i cifrari a blocchi sono metodi per cifrare messaggi di lunghezza arbitraria con cifrati "corti".

Sia F un cifrario a blocchi con lunghezza di blocco n. Sia

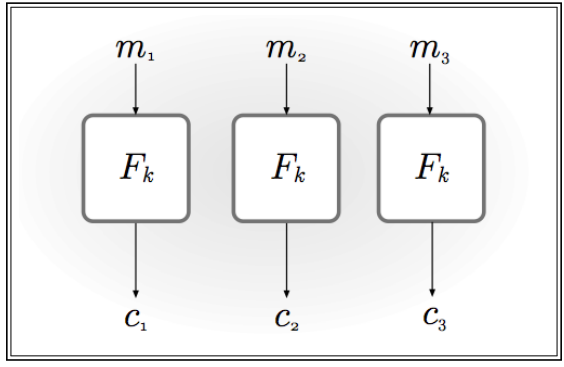
$$m = m_1 m_2 \dots m_\ell, \qquad m_i \in \{0,1\}^n \qquad \text{per ogni } i = 1, \dots, \ell.$$

Se necessario l'ultimo blocco viene "completato" (padded) in modo tale che $|m_\ell| = n$

La modalitá operativa piú semplice é l'Electronic Code Book (ECB) mode

$$c := < F_k(m_1), F_k(m_2), \dots, F_k(m_\ell) >$$

ECB mode



ECB mode - Analisi

ECB é deterministico

- Non puó dare sicurezza CPA
- Non ha neanche cifrature indistinguibili rispetto ad un eavesdropper
  - se un blocco si ripete nel messaggio in chiaro, allora il blocco si ripeterá nel cifrato

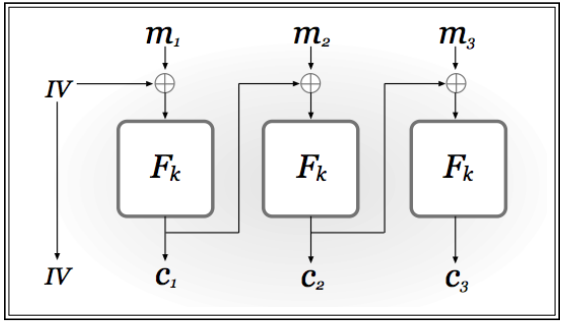
$$m_1 m^* m_3 m_4 m^* \ldots \quad \Rightarrow \quad c_1 c^* c_3 c_4 c^* \ldots$$

É facile allora distinguere tra due messaggi

- Adv sceglie m_0 con un blocco ripetuto ed m_1 con blocchi tutti distinti e analizza i blocchi del cifrato

In conclusione: ECB non dovrebbe mai essere usato

Modalitá Cipher Block Chaining - CBC mode



Viene scelto uniformemente il vettore di inizializzazione $IV \in \{0,1\}^n$. Poi,

$$c_0 = IV, \qquad c_i = F_k(c_{i-1} \oplus m_i), \qquad \text{per } i = 1, \ldots, \ell.$$

Modalitá Cipher Block Chaining - CBC mode

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$m_i = F_k^{-1}(c_i) \oplus c_{i-1}, \qquad \text{per } i = 1, \ldots, \ell.$$

CBC é una modalitá *probabilistica*

Se F é una permutazione pseudocasuale $\Rightarrow$ la modalitá CBC é CPA-sicura.

Inconveniente: la cifratura deve essere effettuata sequenzialmente.

Nota: se il vettore di inizializzazione *IV* *non* viene scelto uniformemente in $\{0,1\}^n$ ma semplicemente in modo che i valori siano distinti

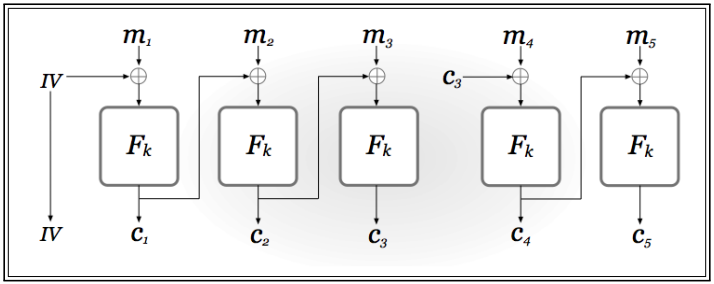
$$\Downarrow$$

la variante *non* é piú sicura.

Chained CBC mode

Una variante di CBC utilizzata é la modalitá con concatenazione.

Idea: invece di usare un nuovo *IV* per ciascuna cifratura, usa *l'ultimo blocco* c_ℓ del cifrato precedente come *IV*.



Chained CBC mode - Analisi

É una variante con stato.

Usata in SSL 3.0 e in TLS 1.0.

Sembra *tanto* sicura *quanto* CBC.

Purtroppo **non** é cosí!

É vulnerabile ad attacchi di tipo chosen-plaintext.

L'attacco si basa sul fatto che il vettore *IV* é noto *prima* che la seconda cifratura abbia luogo.

Chained CBC mode - Analisi

Adv sa che $m_1 \in \{m_1^0, m_1^1\}$

- Osserva il primo cifrato $c :=< IV, c_1, c_2, c_3 >$ di $m_1 m_2 m_3$
- Chiede una cifratura all'oracolo di $m_4 m_5$ dove $m_4 = IV \oplus m_1^0 \oplus c_3$
- Ottiene $< c_4, c_5 >$
- Se $c_4 = c_1$ allora dá in output $b' = 0$ (i.e., $m_1 = m_1^0$). Altrimenti, $b' = 1$.

É facile verificare che $c_4 = c_1$ se e solo se $m_1 = m_1^0$. Infatti:

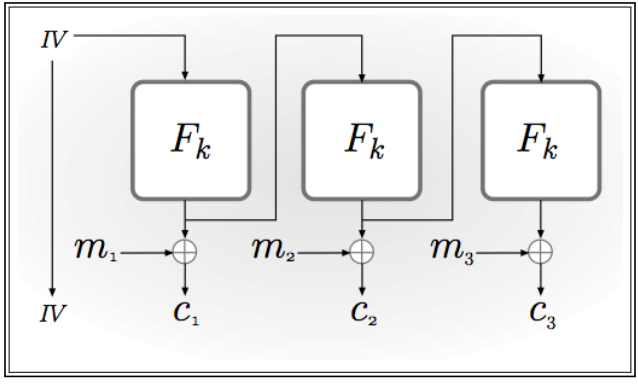
$$c_4 = F_k(m_4 \oplus c_3) = F_k((IV \oplus m_1^0 \oplus c_3) \oplus c_3) = F_k(IV \oplus m_1^0) \quad \text{mentre} \quad c_1 = F_k(IV \oplus m_1).$$

Chained CBC mode - Analisi

Lezione: ogni modifica ad uno schema crittografico puó essere pericolosa, anche se questa modifica sembra ragionevole ed innocua.

Modalitá Output Feedback Mode - OFB mode

Realizza uno stream cipher asincrono



Modalitá Output Feedback Mode - OFB mode

Viene scelto uniformemente il vettore di inizializzazione $IV \in \{0,1\}^n$.

Si genera la stringa pseudocasuale

$$y_0 = IV, \qquad y_i = F_k(y_{i-1}), \qquad \text{per } i = 1, \ldots, \ell.$$

Ciascun blocco m_i del messaggio in chiaro viene cifrato calcolando

$$c_0 = y_0, \qquad c_i = m_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$y_0 = c_0, \qquad y_i = F_k(y_{i-1}), \quad \text{e} \qquad m_i = c_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

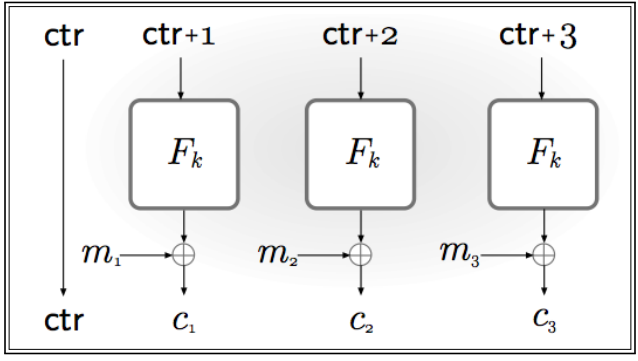
Modalitá Output Feedback Mode - OFB mode

Osservazioni:

- Non é necessario che F sia invertibile (neanche che sia una permutazione).
- Non é necessario che il messaggio abbia una lunghezza multipla di n. La stringa pseudocasuale puó essere troncata dove serve.
- La variante con stato (senza IV per cifrature successive) é sicura. Equivale ad uno stream cipher sincronizzato.
- La modalitá OFB é CPA-sicura se F é una funzione pseudocasuale.
- Il "pad" pseudocasuale puó essere pre-computato.

Modalitá Counter Mode - CRT mode

Puó essere vista come uno stream cipher asincrono costruito a partire da un cifrario a blocchi.



Modalitá Counter Mode - CRT mode

Viene scelto uniformemente $crt \in \{0,1\}^n$.

Si procede poi come segue

$$y_0 = crt, \qquad y_i = F_k((crt + i) \bmod 2^n), \qquad \text{per } i = 1, \ldots, \ell.$$

Ciascun blocco m_i del messaggio in chiaro viene cifrato calcolando

$$c_0 = y_0, \qquad c_i = m_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

La decifratura del cifrato $< c_0, c_1, \ldots, c_\ell >$ viene effettuata calcolando

$$crt = c_0, \qquad y_i = F_k((crt + i) \bmod 2^n), \quad \text{e} \quad m_i = c_i \oplus y_i, \qquad \text{per } i = 1, \ldots, \ell.$$

CRT mode - Analisi

Osservazioni

- La decifratura non richiede che F sia invertibile (né tantomeno che sia una permutazione)
- Lo stream generato puó essere troncato alla lunghezza del messaggio in chiaro e puó essere generato in anticipo.
- La variante "con stato" della modalitá CRT é sicura.
- CRT puó essere *parallelizzata*.
- La decifratura puó essere selettiva: decifrare direttamente l'i-esimo blocco del cifrato.