Introduzione

Capitolo 1

Il Modello Computazionale

1.1 L'avversario

Il tipico avversario con cui si ha a che fare quando si studiano cifrari o protocolli crittografici nel modello computazionale, è un avversario con risorse di calcolo *limitate*. Limitate nel senso che si sceglie di porre un limite alla potenza di calcolo dell'avversario. Questo significa che: non avremo a che fare con un avversario che ha una potenza computazionale infinita o un tempo illimitato a disposizione. Sebbene siano stati ideati cifrari sicuri anche rispetto ad avversari non limitati[1], questi hanno alcuni difetti: come per esempio il fatto che la chiave debba essere lunga quanto il messaggio o che sia utilizzabile una sola volta. Per rappresentare in modo formale un avversario con risorse di calcolo limitate, lo si può pensare come un algoritmo appartenente ad una particolare classe di complessità computazionale[2].
Da sempre si considerano efficienti gli algoritmi che terminano in un numero di passi polinomiale nella lunghezza dell'input, mentre si considerano inefficienti quelli che hanno una complessità computazionale maggiore. Può sembrare quindi naturale immaginare gli avversari come degli algoritmi che terminano in un numero polinomiale di passi rispetto alla lunghezza dell'input. Come si può notare non si fa nessuna assunzione particolare sul comportamento dell'avversario. Le uniche cose che sappiamo, sono che:

- l'avversario non conosce la chiave, ma conosce l'algoritmo di cifratura utilizzato e i parametri di sicurezza, come per esempio la lunghezza della chiave[3].

[1] one-time pad ne è un esempio lampante.

[2] un avversario è alla fine dei conti una macchina di Turing che esegue un algoritmo.

[3] Un famoso principio della crittografia afferma infatti che l'algoritmo di cifratura non deve essere segreto e deve poter cadere nelle mani del nemico senza inconvenienti.

- l'avversario vuole essere efficiente, ovvero polinomiale.

Non si fanno ipotesi sull'algoritmo che questo andrà ad eseguire. Per esempio dato un messaggio cifrato $c = E_k(m)$, non ci aspettiamo che l'avversario non decida di utilizzare la stringa c' tale che: $c' = D_{k'}(E_k(m))$ con $k \neq k'$. Ovvero: sarebbe sbagliato supporre che l'avversario non cerchi di decifrare un messaggio mediante una chiave diversa da quella utilizzata per cifrarlo. Nel modello computazionale i messaggi sono stringhe di bit e l'avversario può effettuare qualsiasi operazione su queste. Questa visione è, a differenza di quella che si ha nel modello formale, sicuramente molto più realistica[Dwo06].

Non bisogna però dimenticare che un avversario può sempre *indovinare* il segreto che cerchiamo di nascondere, o che cifriamo. Per esempio: se il segreto che si cerca di nascondere ha una lunghezza di n bit, l'avversario può sempre lanciare una moneta n volte e associare, via via, la testa della moneta al valore 1 e la croce al valore 0. La probabilità che l'avversario ottenga una stringa uguale al segreto è ovviamente di $\frac{1}{2^n}$. Questa probabilità tende a 0 in modo esponenziale al crescere della lunghezza del segreto, ma per valori finiti di n questa probabilità non sarà mai 0. È quindi più realistico cercare di rappresentare l'avversario come un algoritmo che, oltre a terminare in tempo polinomiale, ha anche la possibilità di effettuare scelte random. La classe dei problemi risolti da questo tipo di algoritmi è indicata con la sigla *BPP* (i.e. *Bounded-Probability Polynomial Time*).
Un modo più formale di vedere questo tipo di algoritmi è il seguente: si suppone che la macchina di Turing che esegue l'algoritmo, oltre a ricevere l'input, diciamo x, riceve un input ausiliario r. Questa stringa di bit r, rappresenta una possibile sequenza di lanci di moneta. Quando la macchina dovrà effettuare una scelta random, non dovrà far altro che prendere il successivo bit dalla stringa r, e prendere una decisione in base ad esso (è, in effetti, come se avesse preso una decisione lanciando una moneta). Ecco quindi che il nostro tipico avversario si configura come un algoritmo polinomiale probabilistico. È inoltre giustificato cercare di rendere sicuri[4] gli schemi crittografici rispetto, principalmente, a questo tipo di avversario. Con questa scelta si cerca di rispettare il più possibile un famoso principio di Kerckhoffs[5] che afferma:

> *Un cifrario deve essere, se non matematicamente, almeno praticamente indecifrabile.*

[4] In qualsiasi modo si possa intendere il concetto di sicurezza. Vedremo che in seguito si daranno delle definizioni rigorose di questo concetto.

[5] Auguste Kerckhoffs (19 Gennaio 1835 – 9 Agosto 1903) fu un linguista olandese e un famoso crittografo.

Non è quindi necessario dimostrare che un particolare schema crittografico sia inviolabile, ma basta dimostrare che:

- in tempi ragionevoli lo si può violare solo con scarsissima probabilità
- lo si può violare con alta probabilità ma solo in tempi non ragionevoli

Sappiamo che il concetto di *tempo ragionevole* è catturato dalla classe degli algoritmi polinomiali probabilistici. Vediamo ora di catturare il concetto di *scarsa probabilità*.

1.2 Funzioni trascurabili e non ...

In crittografia i concetti di *scarsa probabilità* e di evento *raro* vengono formalizzati attraverso la nozione di funzione trascurabile.

Definizione 1.1 *Funzione Trascurabile (negligible). Sia* $\mu : \mathbb{N} \to \mathbb{R}^+$ *una funzione. Si dice che* μ *è trascurabile se e solo se per ogni polinomio p, esiste* $C \in \mathbb{N}$ *tale che* $\forall n > C$*:* $\mu(n) < \frac{1}{p(n)}$.

Una funzione trascurabile quindi, è una funzione che tende a 0 in modo più veloce dell'inverso di qualsiasi polinomio. Un'altra definizione utile è la seguente:

Definizione 1.2 *Funzione Distinguibile (noticeable). Sia* $\mu : \mathbb{N} \to \mathbb{R}^+$ *una funzione. Si dice che* μ *è distinguibile se e solo se esiste un polinomio p, tale per cui esiste* $C \in \mathbb{N}$ *tale che* $\forall n > C$*:* $\mu(n) > \frac{1}{p(n)}$.

Per esempio la funzione $n \to 2^{-\sqrt{n}}$ è una funzione trascurabile, mentre la funzione $n \to \frac{1}{n^2}$ non lo è. Ovviamente esistono anche funzioni che non sono né trascurabili né distinguibili. Per esempio, la seguente funzione definita per casi: $f(n) = \begin{cases} 1, & \text{se } n \text{ è pari} \\ 0, & \text{se } n \text{ è dispari} \end{cases}$ non è né trascurabile né distinguibile. Questo perchè le definizioni precedenti, pur essendo molto legate, non sono l'una la negazione dell'altra.

Se sappiamo che in un esperimento un evento avviene con una probabiltà trascurabile, quest'evento si verificherà con una probabilità trascurabile anche se l'esperimento viene ripetuto molte volte (ma sempre un numero polinomiale di volte), e quindi per la legge dei grandi numeri, con una frequenza altrettanto trascurabile[6]. Le funzioni trascurabili infatti, godono di due particolari proprietà di chiusura, enunciate nella seguente:

[6]In modo informale, la legge debole dei grandi numeri afferma che: per un numero grande di prove, la frequenza approssima la probabilità di un evento.

Proposizione 1.1 *Siano μ_1, μ_2 due funzioni trascurabili e sia p un polinomio. Se $\mu_3 = \mu_1 + \mu_2$, e $\mu_4 = p \cdot \mu_1$, allora μ_3, μ_4 sono funzioni trascurabili.*

Se quindi, in un esperimento, un evento avviene solo con probabilità trascurabile, ci aspettiamo che, anche se ripetiamo l'esperimento un numero polinomiale di volte, questa probabilità rimanga comunque trascurabile.
Per esempio: supponiamo di avere un dado truccato in modo che la probabilità di ottenere 1 sia trascurabile. Allora se lanciamo il dado un numero polinomiale di volte, la probabilità che esca 1 rimane comunque trascurabile.
È ora importantissimo notare che: **gli eventi che avvengono con una probabilità trascurabile possono essere ignorati per fini pratici**. In [KL07], infatti leggiamo:

> *Events that occur with negligible probability are so unlikely to occur that can be ignored for all practical purposes. Therefore, a break of a cryptographic scheme that occurs with negligible probability is not significant.*

Potrebbe sembrare pericoloso utilizzare degli schemi crittografici che ammettono di essere violati con probabilità trascurabile. Ma questa possiblità è cosi remota, che se ci preoccupassimo, allora per amor di coerenza, dovremmo anche essere ragionevolmente sicuri di fare sei all'enalotto giocando una schedina semplice. Finora abbiamo parlato sempre di funzioni che prendono in input un argomento non meglio specificato. Al crescere di questo parametro, le funzioni si comportano in modo diverso a seconda che siano trascurabili, oppure no. Ma cosa rappresenta nella realtà questo input? In genere questo valore rappresenta un generico parametro di sicurezza, indipendente dal segreto. Di solito lo si pensa come la lunghezza in bit delle chiavi.
Tutte le definizioni di sicurezza che vengono date nel modello computazionale e che utilizzano le probabilità trascurabili, sono di tipo *asintotico*. Un template di definizione di sicurezza è il seguente [KL07]:

> *A scheme is secure if for every probablistic polynomial-time adversary $\boldsymbol{A}$ [...], the probability that $\boldsymbol{A}$ succeds in this attack [...] is negligible*

Essendo questo schema di definizione asintotico (nel parametro di sicurezza, che indicheremo con n d'ora in poi), è ovvio che non considera valori piccoli di n. Quindi: se si dimostra che un particolare schema crittografico è sicuro secondo una definizione di questo tipo, può benissimo capitare che per valori piccoli di n lo schema sia violabile con alta probablità e in tempi ragionevoli.

1.3 Indistinguibilità Computazionale

Se due oggetti, sebbene profondamente diversi fra loro, non possono essere distinti, allora sono da un certo punto di vista equivalenti. Nel caso della crittografia computazionale due oggetti sono computazionalmente equivalenti se nessuna algoritmo efficiente li può distinguere. Possiamo immaginare che un algoritmo riesca a distinguere due oggetti, se quando gli si da in input il primo, lui da in output una costante c, mentre se gli si fornisce come input il secondo da in output una costante c' e ovviamente $c \neq c'$. La definizione tipica di indistinguibilità computazionale è data prendendo come oggetti da distinguere delle particolari distribuzioni statistiche detti *ensembles*.

Definizione 1.3 *Ensemble. Sia I un insieme numerabile infinito.*
$X = \{X_i\}_{i \in I}$ è un ensemble su I se e solo se è una sequenza di variabili statistiche, ognuna con una distribuzione di probabilità uniforme.

Un ensemble è quindi una sequenza infinita di distribuzioni di probabilità[7]. Tipicamente le variabili dell'ensemble sono stringhe di lunghezza i. X_i è quindi la distribuzione di probabilità uniforme su stringhe di lunghezza i.

Ora supponiamo di avere due ensemble X e Y. Intuitivamente queste distribuzioni sono indistinguibili se nessun algoritmo (efficiente) può accettare infiniti elementi di X_n (ovvero stampando 1 su input preso da X_n) e scartare infiniti elementi di Y_n (ovvero stampare 0 su input preso da Y_n). È importante notare che sarebbe facile distinguere due *singole* distribuzioni usando un approccio esaustivo, ecco perchè si considerano sequenze infinite di ditribuzioni finite. In poche parole questi ensemble sono indistinguibile se ogni algoritmo accetta $x \in X_n$ se e solo se accetta $y \in Y_n$. Ovviamente il *se e solo se* non può e non deve essere inteso in senso *classico* ma deve essere inteso in senso statistico. Dopo questa breve introduzione all'indistinguibilità siamo pronti per dare una definizione rigorosa:

Definizione 1.4 *Indistinguibilità computazionale. Due ensemble $X = \{X_n\}$, $Y = \{Y_n\}$ sono computazionalmente indistinguibili se e solo se per ogni algoritmo $D \in BPP$ (detto distinguitore) esiste μ trascurabile tale che: $|Pr[D(1^n, X_n) = 1] - Pr[D(1^n, Y_n) = 1]| \leq \mu(n)$.*

Dove $Pr[D(1^n, X_n) = 1]$ è la probabilità che, scegliendo x secondo la distribuzione X_n e fornendo questo valore al distinguitore insieme al valore

[7] siccome si parla di distribuzioni su stringhe di bit con lunghezza finita, in crittografia computazionale si considerano ensemble che sono una sequenza inifinta di distribuzioni finite di stringhe di bit.

1^n, il distinguitore stampi 1. Il fatto che al distinguitore si fornisca anche il valore del parametro di sicurezza in base unaria, serve ad esser sicuri che in ogni caso il distinguitore impieghi un tempo polinomiale nel parametro di sicurezza. Infatti il distinguitore quando si troverà a dover leggere il primo parametro, necessariamente impiegherà un tempo polinomiale nel parametro di sicurezza, visto che questo è stato fornito in base unaria.

La definizione di indistinguibilità computazionale cattura quindi il seguente concetto: se due ensemble sono computazionalmente indistinguibili, allora la probabilità che un distinguitore riesca a discernere i valori provenienti da un insieme rispetto all'altro è trascurabile; di conseguenza agli occhi del distinguitore gli ensemble non sono differenti e quindi sono per lui equivalenti (o meglio computazionalmente equivalenti). Non è raro, nell'ambito scientifico in particolare, basarsi sul concetto generale di indistinguibilità al fine di creare nuove classi di equivalenza di oggetti.

> *The concept of efficient computation leads naturally to a new kind of equivalence netween objects: Objects are considered to be computatinally equivalent if they cannot be differentiated by any efficient procedure.* We note that considering indistinguishable objects as equivalent is one of the basics paradigms of both science and real-life situations. Hence, we believe that the notion of computational indistinguishability is a very natural one.[Gol00]

1.4 Pseudocasualità e generatori pseudocasuali

Argomento centrale di questa sezione è il concetto di *pseudocasualità* (pseudorandomness), applicato a stringhe di bit di lunghezza finita. Parlare di pseudocasualità applicata ad una *singola* stringa ha poco senso quanto ne ha poco parlare di singola stringa casuale (random). Infatti il concetto di casualità (come quello di pseudocasualità) si applica a distribuzioni di oggetti (stringhe di bit nel nostro caso). In crittografia, la nozione di casualità è fortemente legata a quella di distribuzione uniforme. Un insieme discreto di oggetti è caratterizzato da una distribuzione uniforme se la probabilità è equamente distribuita su tutti gli oggetti e quindi *l'estrazione* di un elemento è del tutto casuale, perchè non ci sono elementi più probabili di altri. In pratica esistono vari modi per ottenere bits random. Per esempio si possono utilizzare dei software basati sulle interazioni, considerate imprevedibili, che l'utente ha con la macchina: velocità con cui preme i tasti della tastiera, movimenti del mouse ecc, ecc ...

Il concetto di pseudorandomness è un caso particolare di indistinguibilità, infatti una distribuzione è *pseudorandom* se nessuna procedura efficiente, può distinguerla dalla distribuzione uniforme.

Definizione 1.5 *Pseudoarandomness. Sia* $U = \{U_{l(n)}\}_{n\in\mathbb{N}}$ *la distribuzione uniforme sull'insieme di stringhe* $\{0,1\}^{l(n)}$. *L'ensemble* $X = \{X_n\}_{n\in\mathbb{N}}$ *è detto pseudorandom se e solo se è computazionalmente indistinguibile da dall'ensemble* U[8].

Data questa definizione, possiamo finalmente definire formalmente cosa sia un generatore pseudorandom.

Definizione 1.6 *Generatore Pseudorandom. Sia* $l : \mathbb{N} \to \mathbb{N}$ *un polinomio. Sia* G *un algoritmo polinomiale deterministico tale che* $\forall s \in \{0,1\}^n$ $G(s) \in \{0,1\}^{l(n)}$. *Allora* G *è un generatore pseudorandom se e solo se valgono le seguenti condizioni:*

- *(Espansione)* $\forall n : l(n) > n$
- *(Pseudocasualità)* $\forall D \in BPP, \exists \mu$ *trascurabile tale che* $|Pr[D(1^{l(n)}, r) = 1] - Pr[D(1^{l(n)}, G(s)) = 1]| \leq \mu(n)$ *con* $r \in U_{l(n)}$ *e* $s \in U_n$

1.5 Dimostrazioni Basate su Games

[8]Con la notazione A^k dove A è un insieme di simboli e k un numero, si intende l'insieme di tutte le stringhe di lunghezza k ottenibili con simboli di A.

Capitolo 2

CryptoVerif

Capitolo 3

Risultati Raggiunti

Capitolo 4

Conclusioni

Bibliografia

[Dwo06] Cynthia Dwork, editor. *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*. Springer, 2006.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.