Impossibilità computazionale di un attacco

Finora abbiamo usato i termini di **calcolo facile** e di **calcolo difficile** con il loro significato più intuitivo. Ciò ci ha consento di differenziare nettamente le attività computazionali richieste a chi usa legittimamente un sistema informatico sicuro, da quelle che deve invece mettere in atto chi vuole attaccarne la sicurezza.

E' però utile avere a disposizione anche una definizione rigorosa di tali termini: a tal fine si deve far riferimento alla Teoria degli algoritmi e, più in particolare, a quella branca denominata **Teoria della complessità computazionale** [8].

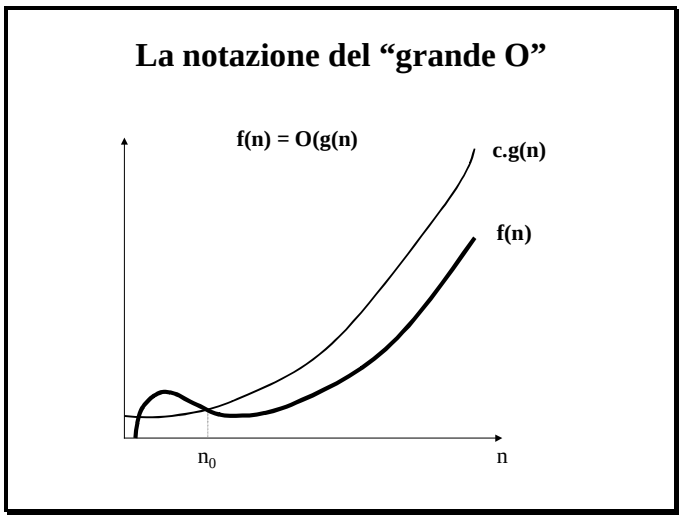
In questa sede ci è però sufficiente richiamare soltanto alcune definizioni, potendo rinviare ogni necessario approfondimento allo studio della disciplina di Ricerca Operativa.

Glossario di Teoria della complessità computazionale

Algoritmo	Procedimento che risolve un ben definito problema computazionale, prendendo un insieme di valori come dato di *input* e producendo un insieme di valori come *output*.
Dimensione dell'input	Numero **n** di bit che rappresentano il dato d'ingresso (*input size*).
Complessità computazionale di un algoritmo	Quantità di risorse di calcolo necessarie per eseguirlo: risorse tipicamente prese in considerazione sono il tempo di esecuzione *(running time)*, la capacità di memoria *(memory size)*, il numero e la dimensione degli stack.
Tempo di esecuzione di un algoritmo	Numero **N** di operazioni elementari, o di passi, che l'algoritmo deve eseguire per terminare. In generale **N** dipende a sua volta dalla dimensione **n** dell'input: in alcuni casi **N = f(n)** è esprimibile analiticamente, in altri no. A parità di **n** si riscontrano però di solito valori diversi di **N**.
Tempo di esecuzione nel caso peggiore	Il più grande valore di **N** per qualsiasi input di dimensione **n**.
Ordine di grandezza del tempo di esecuzione	Modalità di incremento di **N** all'aumentare senza limiti di **n**. Se **f(n)** è espressa da una formula**,** si considera soltanto il termine **g(n)** che cresce più rapidamente con **n**; in caso contrario si considera una funzione **g(n)** ed una costante **c** tale che $0 \le f(n) \le c.\ g(n)$ per ogni $n \ge n_0$
T = O(g(n))	Notazione simbolica dell'ordine di grandezza del tempo di esecuzione.
Algoritmo polinomiale	Algoritmo con **g(n)** polinomiale in **n.** L'ordine di grandezza del tempo di esecuzione è indicato con $T = O(n^t)$, ove **t** è l'esponente più grande presente nel polinomio.
Algoritmo esponenziale	Algoritmo con **g(n)** esponenziale in **n**. L'ordine di grandezza del tempo di esecuzione è indicato con **T = O(exp (n))**.
Problema facile	Problema, detto anche di tipo **P**, per il quale esiste un algoritmo polinomiale in grado di risolverlo su una **macchina di Turing deterministica**.
Problema difficile	Problema per cui non sono stati fino ad ora individuati, e probabilmente non saranno mai individuati, algoritmi di risoluzione con tempo polinomiale.

Prima di impiegare queste definizioni nel contesto della sicurezza informatica occorre un momento di riflessione.

[8] vedi S. Martello: "Lezioni di Ricerca Operativa" Esculapio 2001, [4] cap.3 e T.H. Cormen, C.E. Leiserson, R.L. Rivest "Introduzione agli algoritmi", vol.1, cap.2, *Jackson Libri* 1994



La valutazione del tempo di esecuzione nel **caso peggiore** è limitante, quando si studiano algoritmi di attacco: occorre, infatti, sapersi difendere dal **caso migliore** (quello più agevole per l'intruso) ed è quindi necessario adottare ogni provvedimento atto ad impedire che l'avversario si trovi di fronte ad **istanze del problema** di facile soluzione.

La notazione del "**grande o**" si limita inoltre ad evidenziare solo come si incrementa il tempo di esecuzione dell'algoritmo al crescere senza limiti della dimensione del input.

Per ottenere sicurezza è dunque necessario saper individuare bene il valore di **n** al di sopra del quale il calcolo diventa realmente impossibile.

Una volta chiariti i limiti, è giusto evidenziare il contributo positivo che la Teoria della complessità può dare alla valutazione della sicurezza di un meccanismo crittografico. E' sufficiente, infatti, rispettare due sole regole.

- R10: "*ogni algoritmo che difende una proprietà critica dell'informazione deve avere tempo **polinomiale***".

ESEMPI - Le notazioni **O(1)**, **O(n)**, **O(n^3)** caratterizzano algoritmi polinomiali con crescente tempo d'esecuzione

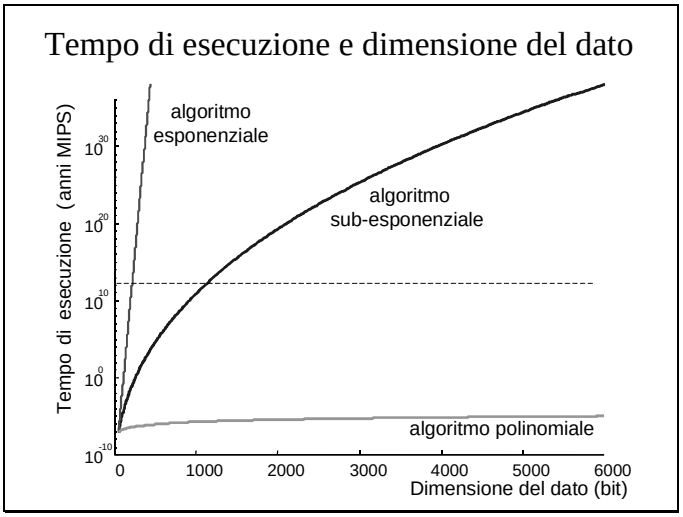
- R11: "*ogni algoritmo che attacca una proprietà critica dell'informazione deve avere tempo **esponenziale***".

ESEMPI - **O(exp (n))**, **O(exp ($n^{1/2}$))** caratterizzano algoritmi esponenziali con tempo d'esecuzione decrescente

E' bene mettere fin d'ora in evidenza che la Crittanalisi considera **difficili** anche algoritmi di attacco, detti **sub-esponenziali**, il cui tempo d'esecuzione ha una componente polinomiale ed una esponenziale:

O(exp ($(n)^{\alpha}$ $(\ln(n)^{1-\alpha})$) con $0 < \alpha < 1$.

A parità di dimensione dell'input, la loro difficoltà di calcolo è decisamente più bassa di quella di un algoritmo esponenziale, pur restando incomparabilmente più alta di quella di un algoritmo polinomiale.



Le curve di figura intendono dare un'idea qualitativa sulle modalità di crescita del tempo di esecuzione degli algoritmi polinomiali, sub-esponenziali ed esponenziali.

Per individuare la dimensione di input che determina l'**impossibilità computazionale**, degli attacchi, le tecnologie per la sicurezza hanno assunto come riferimento un calcolatore in grado di eseguire **un milione di operazioni al secondo** o, più brevemente, **MIPS**.

Il tempo di esecuzione di un attacco è di conseguenza espresso in **anni MIPS** ed indica quindi il numero di anni impiegati dal calcolatore di riferimento per portarlo a termine.

Negli anni '90 si è valutato che **10^{12}** anni MIPS fossero adeguati per garantire l'impossibilità di calcolo.

La **legge di Moore** (a parità di costo, circa ogni 18 mesi è realizzabile un calcolatore con potenza doppia del precedente) e la crescente possibilità di impiegare il calcolo parallelo (cioè di scomporre l'algoritmo in parti eseguibili contemporaneamente e di assegnare tale carico di lavoro a migliaia di macchine connesse in rete o a sistemi contenenti migliaia di processori) continueranno a spingere questo limite sempre più in alto. Il dato oggetto di un attacco dovrà di conseguenza essere rappresentato da una stringa di bit sempre più lunga.

Nella valutazione della sicurezza si può però fare anche a meno degli anni MIPS, introducendo il concetto di **livello di sicurezza**. Questa diversa unità di misura della robustezza assume come riferimento l'algoritmo di **ricerca esauriente** di un numero formato da **n** bit e dotato di una certa proprietà **P**.

L'algoritmo di ricerca esauriente (in Crittografia è detto **attacco con forza bruta**) prevede di provare uno dopo l'altro tutti i valori espressi con n bit, controllando ogni volta se la proprietà P è soddisfatta o meno. Nel caso peggiore l'algoritmo termina dopo $\mathbf{2^n}$ passi ed ha quindi complessità **O(exp(n))**.

Consideriamo ora una certa proprietà critica dell'informazione per la quale, in un certo sistema, è stata predisposta una certa difesa crittografica e supponiamo di sapere che il più efficace algoritmo di attacco richiede almeno $\mathbf{2^n}$ passi per terminare.

In queste condizioni si dice che il sistema ha **un livello di sicurezza di n bit**, cioè che è attaccabile solo con uno sforzo computazionale pari a quello richiesto dalla ricerca esauriente di un numero di n bit.

Il vantaggio di impiegare il livello di sicurezza al posto degli anni MIPS è quello di non fissare la velocità del calcolatore che esegue l'attacco e quindi di prescindere dall'evoluzione tecnologia. Naturalmente è possibile passare dalla misura in anni MIPS a quella in livello di sicurezza e viceversa

ESEMPIO - $\mathbf{10^{12}}$ anni MIPS corrispondono ad un livello di sicurezza di 85 bit.

E' opinione largamente condivisa che i meccanismi progettati ed impiegati attualmente devono avere un livello di sicurezza il più possibile vicino a **128 bit**.