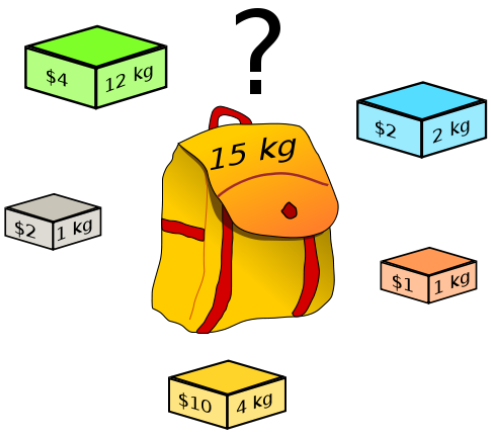
idea della crittografia a chiave pubblica

- sviluppare un crittosistema in cui data la funzione di cifratura e_k sia computazionalmente difficile determinare d_k
- Bob rende pubblica la sua funzione di cifratura e_k
- Alice (e chiunque altro) può scrivere a Bob, cifrando il messaggio con la e_k senza bisogno di accordi preliminari
- Bob è l'unico che può decifrare il messaggio
  - solo Bob ha un informazione supplementare che permette la decifratura
  - ci sarà una chiave pubblica nota a tutti – che serve per cifrare
  - e una chiave privata nota solo a Bob – che serve per decifrare

- sia **P** la classe dei problemi P per cui esiste un algoritmo che risolve P in tempo polinomiale (esempio trovare il massimo comun divisore di due numeri interi)
- il problema P appartiene alla classe **NP** se esistono algoritmi (non necessariamente polinomiali) che risolvono il problema ma è possibile verificare, con un algoritmo polinomiale, se un dato assegnato è soluzione o meno del problema (esempio: fattorizzare un intero n)
- i problemi **NP**–completi sono i problemi computazionalmente più difficili tra quelli di tipo

problema dello zaino (knapsack problem)

- un problema **NP**–completo è il problema dello zaino
- dato uno zaino che può contenere b unità
- e dati k oggetti di volume $a_1, \ldots, a_k$
- il problema è riempire lo zaino usando alcuni oggetti della lista



problema dello zaino (knapsack problem)

- dobbiamo trovare una k-pla $(e_1, \ldots, e_k)$, $e_i = 0, 1$
- tale che $b = \sum_i e_i a_i$
- se possibile – oppure mostare che una tale k-pla non esiste
- il problema è in **NP** – data una k-pla, è facile verificare se è una soluzione
- ma trovare una soluzione è difficile – procedendo per tentativi ho 2^k possibilità da controllare
- e non si conosce un algoritmo veloce per risolverlo – il problema dello zaino è **NP**–completo

esempio

- data la lista $(323, 412, 33, 389, 544, 297, 360, 486)$
- e dato $b = 1228$
- un metodo possibile è un procedimento greedy (avido, goloso)
- a ogni passo, prendo l'oggetto più grande che entra ancora nello zaino
- in questo esempio, ho
  $1228 = 544 + 684 = 544 + 486 + 198 = 544 + 486 + 33 + 165$
- e qui ci fermiamo
- invece si ha che $1228 = 412 + 33 + 297 + 486$
- se avessimo una lista di 100 numeri di 50 cifre ognuno il problema diventa inavvicinabile

un crittosistema dal problema dello zaino

- la chiave pubblica di Bob è una lista $(a_1, \ldots, a_k)$ di interi positivi
- per cifrare, il messaggio dev'essere una stringa binaria di lunghezza k
- $x = (e_1, \ldots, e_k)$, $e_i = 0, 1$ – Alice calcola $\sum_i e_i a_i = b$
- e trasmette b a Bob
- nessun attaccante può decifrare – bisogna risolvere un problema dello zaino!
- in questo momento, non può decifrare neanche Bob...

sequenze supercrescenti

- ci sono particolare k-ple per cui il problema dello zaino è facile
- la sequenza $(a_1, \ldots, a_k)$ si dice supercrescente se ogni termine è maggiore della somma dei termini precedenti
- $\sum_{j=1}^{i-1} a_j < a_i$
- se ho una lista $(a_1, \ldots, a_k)$ supercrescente
- è facile vedere che il procedimento greedy visto prima risolve il problema dello zaino
- e che - se esiste - la soluzione è unica
- per esempio la successione $1, 2, 4, 8, 16, \ldots$ delle potenze di due è supercrescente
- il procedimento greedy è quello che si usa per scrivere un intero in base 2

il crittosistema di Merkle–Hellman

- non possiamo usare una sequenza supercrescente come chiave pubblica!
- bisogna "mascherare" questa proprietà
- Bob sceglie una lista $(a_1, \ldots, a_k)$ supercrescente
- sceglie poi un intero $n > \sum a_i$ e un intero u con $(u, n) = 1$
- costruisce una nuova lista $(a_1^*, \ldots, a_k^*)$
- dove $a_i^* = ua_i \pmod{n}$
- $(a_1^*, \ldots, a_k^*)$ è la chiave pubblica
- $(a_1, \ldots, a_k)$, n e u sono la chiave privata

- la cifratura funziona come già detto: Alice deve cifrare $x = (e_1, \ldots, e_k)$ – calcola $\sum_i e_i a_i^* = b^*$ e trasmette b^* a Bob
- per decifrare, Bob calcola v, l'inverso di u modulo n
- poi calcola $vb^* \equiv b \bmod n$
- $b \equiv vb^* = v \sum e_i a_i^* \equiv v \sum e_i u a_i$ modulo n
- $v \sum e_i u a_i = uv \sum e_i a_i \equiv \sum e_i a_i$ modulo n
- ora sia b che $\sum e_i a_i$ sono minori di $n = \sum a_i$
- quindi $b = \sum e_i a_i$
- Bob deve risolvere un caso facile del problema dello zaino per decifrare

esempio

- data la lista supercrescente $(1, 3, 7, 15, 31, 63, 127, 255)$
- Bob sceglie $n = 557$ maggiore della somma dei termini della lista e $u = 323$ coprimo con 557
- moltiplicando per 323 e riducendo mod 557 ottiene $(323, 412, 33, 389, 544, 297, 360, 486)$
- la stringa binaria 01100101 viene codificata da Alice come $412 + 33 + 297 + 486 = 1228$
- Bob riceve 1228 – sa che l'inverso di 323 mod 557 è 169
- calcola $1228 \cdot 169 \bmod 557$ e ottiene 328
- $328 = 255 + 73 = 255 + 63 + 10 = 255 + 63 + 7 + 3$
- riottiene quindi la stringa 01100101

il crittosistema di Merkle–Hellman

- il crittosistema ora descritto è dovuto a Merkle e a Hellman (1978)
- è elegante – e molto più semplice dell'RSA
- sfortunatamente non è molto sicuro
- A. Shamir, A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem, IEEE Transactions on Information Theory (1984)
- il "mascheramento" non funziona molto bene

ordine di un gruppo

- G un gruppo finito: ordine di $G = o(G) =$ numero di elementi di G
- l'insieme degli invertibili di $\mathbb{Z}_n$ è un gruppo rispetto al prodotto (mod n)
- si denota con $U(\mathbb{Z}_n)$ e ha ordine $\phi(n)$
- esempio: $U(\mathbb{Z}_9) = \{1, 2, 4, 5, 7, 8\}$, ha $\phi(9) = 6$ elementi

funzione ϕ di Eulero

- funzione ϕ di Eulero, o funzione toziente
- è definita sugli interi positivi
- $\phi(n)$ è il numero di interi positivi $\leq n$ che sono coprimi con n
- $\phi(n) = |\{k \in \mathbb{N}, \quad k \leq n \quad | \quad (k, n) = 1\}|$
  - se p è primo, $\phi(p) = p - 1$
  - se p è primo, $\phi(p^k) = p^k - p^{k-1}$
  - se n, m sono coprimi, allora $\phi(n \cdot m) = \phi(m)\phi(n)$
- in questo modo, si può calcolare la funzione di Eulero di ogni intero
- purché se ne conosca la fattorizzazione
- se $n = p_1^{\alpha_1} \dots p_s^{\alpha_s}$
- $\phi(n) = \phi(p_1^{\alpha_1}) \dots \phi(p_s^{\alpha_s}) = (p_1^{\alpha_1} - p_1^{\alpha_1 - 1}) \dots (p_s^{\alpha_s} - p_s^{\alpha_s - 1})$