

Knapsack Ciphers

Davide Caracausi

Indice

Indice	2
1. Premesse	3
1.1. Come trattare il testo – Cifrari a blocchi	3
1.2. Crittografia a chiave pubblica	3
1.2.1. Vantaggi dei cifrari a chiave pubblica.....	5
1.2.2. Svantaggi dei cifrari a chiave pubblica	6
2. Knapsack Ciphers	7
2.1. Problema Knapsack (discreto)	7
2.1.1. Complessità.....	7
2.2. Problema Knapsack semplificato – Sequenze super-increasing	7
2.2.1. Algoritmo 1 – Soluzione con sequenze super-increasing.....	8
2.3. Knapsack Ciphers	9
2.3.1. Cifratura – Premesse	9
2.3.2. Cifratura – Algoritmo.....	10
2.3.3. Decifrazione – Premesse	11
2.4. Implementazione pratica	13
2.5. Difetti del Knapsack Ciphers	13
3. Bibliografia	14

1. Premesse¹

1.1. Come trattare il testo – Cifrari a blocchi

In generale, il messaggio da cifrare/decifrare consiste di una sequenza di simboli discreti, ognuno scelto da un insieme finito. Questi simboli possono essere lettere o parole di un linguaggio, livelli di ampiezza di un segnale quantizzato, bit, etc... e ad essi si fa riferimento con il termine generico di **caratteri**, senza tenere conto della loro dimensione. L'insieme di tutti i simboli di uno stesso tipo è detto **alfabeto**. Inoltre, il messaggio originale ha una lunghezza arbitraria, ossia è una sequenza di caratteri di lunghezza variabile e come tale non può essere trattato tutto in una volta. Occorre, dunque, scegliere come operare sul messaggio originale per poter effettuare l'elaborazione. Le strategie che si possono utilizzare sono due:

- 1) suddividere il messaggio originale in blocchi di grandezza uguale ed applicare a ciascuno di essi la funzione di trasformazione; in tal caso si parla di block cipher o cifrario a blocchi;
- 2) effettuare l'elaborazione sui singoli caratteri trattando quindi il testo in chiaro come fosse una stringa di caratteri; si parla, in tal caso, di stream cipher o cifrario a flusso;

In particolare, un **cifrario a blocchi** divide il testo in chiaro in blocchi di una dimensione fissata (dette appunto blocchi) e opera su ogni blocco indipendentemente un blocco alla volta. Un particolare blocco di testo (o carattere) in chiaro viene mappato in un altrettanto blocco di testo (o carattere) cifrato ogni volta che appare nel testo.

Questa breve panoramica sul come trattare il testo è servita a far capire come, malgrado l'input della funzione di trasformazione siano sempre la chiave ed il messaggio, quest'ultimo assuma “forma diversa” a seconda che si tratti di un cifrario a blocchi oppure di uno a flusso.

1.2. Crittografia a chiave pubblica

Consideriamo un insieme di funzioni cifranti $\{E_e : e \in \mathcal{K}\}$ dove \mathcal{K} è l'insieme delle possibili chiavi, e l'insieme delle corrispondenti funzioni decifranti $\{D_d : d \in \mathcal{K}\}$. Supponiamo ora che ogni coppia di funzioni corrispondenti (E_e, D_e) abbia la proprietà che, conoscendo E_e e dato un qualsiasi testo cifrato C , sia “**computazionalmente complesso**” trovare il messaggio in chiaro P tale che $E_e(P) = C$. Questo implica che data la chiave di cifratura e sia infattibile trovare la corrispondente chiave di decifrazione d . E_e può essere vista come una **funzione a senso unico**, cioè “difficilmente” invertibile, con la particolare caratteristica che, avendo a disposizione delle

¹ Parte scritta in collaborazione con Luigi Maniscalco.

informazioni aggiuntive particolari, dette **scorciatoie**, si possa facilmente trovare la funzione inversa D_d . Nel caso di cifrari a chiave pubblica la scorciatoia è costituita proprio dalla chiave di decifrazione, che consente appunto di decifrare il testo cifrato ottenendo così il testo originale. Questo differisce da quanto avviene nei **cifrari a chiave simmetrica**, dove le chiavi di cifratura e decifrazione (e e d rispettivamente) sono esattamente la stessa. Riportando queste assunzioni su una comunicazione tra due parti, si può procedere come illustrato in Figura 1.

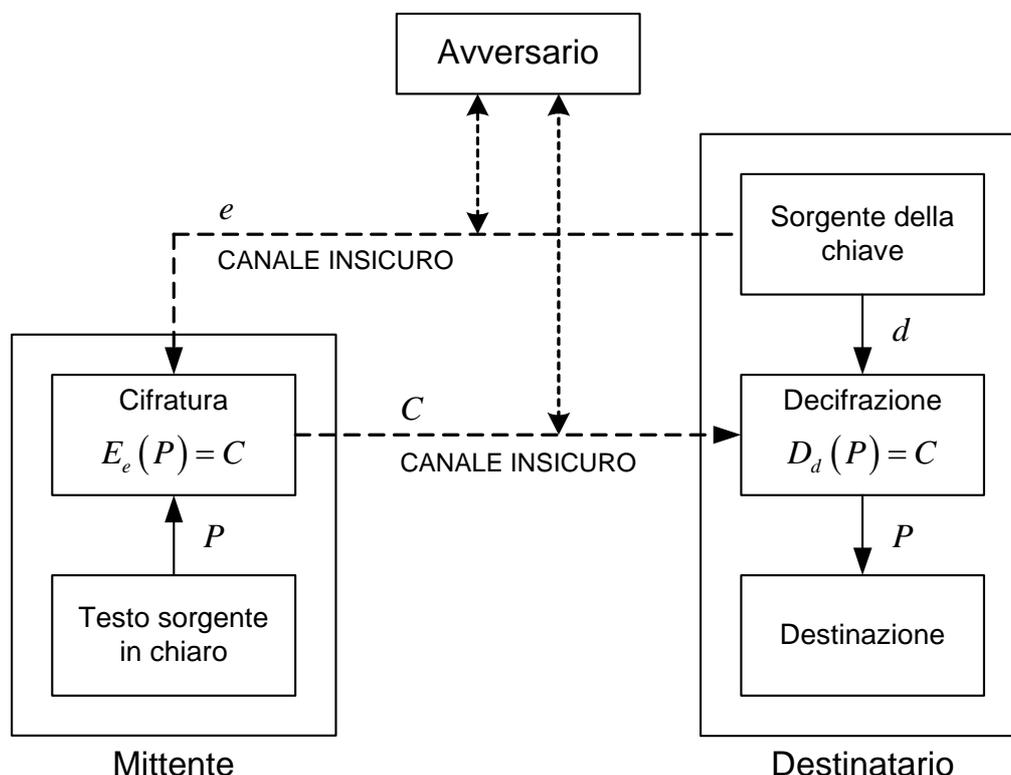


Figura 1 Schema di una comunicazione tra due parti con crittografia a chiave pubblica.

La **procedura** è molto semplice; il destinatario che desidera ricevere il messaggio seleziona una coppia di chiavi (e, d) , quindi manda (ossia rende pubblica) la chiave cifrante e (detta **chiave pubblica**) al mittente del messaggio, ma mantiene segreta la chiave decifrante d (detta **chiave privata**). Il mittente a questo punto può trasformare il testo in chiaro per ottenere il testo cifrato $C = E_e(P)$ e spedirlo al destinatario; quest'ultimo, facendo uso della chiave privata d , può calcolare $P = D_d(C)$ ed ottenere quindi il testo in chiaro.

La particolarità dell'intero processo sta nel fatto che solo il destinatario è in grado di decifrare il messaggio; data infatti la caratteristica della funzione cifrante di essere a senso unico, paradossalmente neppure il mittente, una volta cifrato il messaggio, è più in grado di riottenere il testo in chiaro da quello cifrato senza conoscere l'apposita chiave d .

Ovviamente la comunicazione non viaggia in una sola direzione, pertanto in una **comunicazione a due parti** occorreranno due chiavi pubbliche (una per parte) e due corrispondenti

chiavi private. Estendendo questo concetto a più parti, si necessiterà di tante chiavi pubbliche quante sono le parti in gioco, e occorrerà cifrare ogni messaggio utilizzando la chiave pubblica corrispondente alla parte destinataria del messaggio stesso.

Uno schema di crittografia simile porta al vantaggio di non dover utilizzare un canale sicuro per lo scambio delle chiavi; si possono ad esempio pubblicare sul sito personale o spedirle per posta elettronica. Tuttavia presenta due grossi svantaggi:

1. Le funzioni a senso unico si appoggiano su **problemi matematici** considerati “**computazionalmente intrattabili**”, come ad esempio la scomposizione in fattori primi di numeri interi molto grandi o la risoluzione di problemi NP-completi. Tutti questi problemi hanno il vantaggio di essere facilmente risolvibili se si è a conoscenza di particolari informazioni (le scorciatoie appunto). Diventa quindi “computazionalmente oneroso” effettuare una ricerca esaustiva di tutte le possibili chiavi; i moderni sistemi di crittografia a chiave pubblica sono considerati a questo proposito sicuri, ma non esiste ancora nessuna dimostrazione matematica in grado di garantire la totale assenza di “buchi” in questi sistemi.
2. Lo schema di crittografia a chiave pubblica si presta molto bene agli attacchi di tipo “**uomo nel mezzo**”, dove l’avversario si interpone tra le due parti assumendo l’identità di entrambe, facendo così credere a ciascuna delle parti di comunicare con la controparte, in maniera completamente trasparente. L’unico modo di ovviare un simile inconveniente è quello di appoggiarsi a sistemi di autenticazione dell’entità e della provenienza dei dati, che diventano in questo caso necessari.

Nonostante ciò la neonata crittografia a chiave pubblica (nasce infatti a metà degli anni ’70), gioca oggi un ruolo fondamentale nei sistemi crittografici moderni, garantendo un **buon livello di sicurezza senza la necessità di costosi canali sicuri**. Risulta infatti uno strumento efficace in grado di fornire non solo riservatezza, ma anche delle buone basi per schemi di firme digitali, certificati, e meccanismi di accordo sulle chiavi.

Volendo riassumere:

1.2.1. Vantaggi dei cifrari a chiave pubblica

1. Solo la chiave privata deve essere mantenuta segreta (l’autenticità delle chiavi pubbliche deve comunque essere garantita);
2. A seconda del metodo di utilizzo, una coppia costituita da chiave privata/chiave pubblica può rimanere invariata per considerevoli periodi di tempo (anche diversi anni);
3. Parecchi schemi a chiave pubblica sono in grado di fornire meccanismi di firme digitali relativamente efficienti. Le chiavi utilizzate per descrivere la funzione pubblica di verifica sono tipicamente più piccole di quelle utilizzate per le funzioni basate su cifrari a chiave simmetrica;

1.2.2. Svantaggi dei cifrari a chiave pubblica

1. Le velocità di codifica raggiunte dai più diffusi metodi di cifratura a chiave pubblica sono di diversi ordini di grandezza più piccole di quelle raggiunte dai più conosciuti schemi a chiave simmetrica;
2. Non è ancora stata provata l'assoluta sicurezza di alcuno schema di cifratura a chiave pubblica. Attualmente gli schemi di cifratura a chiave pubblica più efficaci basano la loro sicurezza sulla "presunta" difficoltà di risoluzione di un piccolo insieme di problemi numerici;
3. La crittografia a chiave pubblica non ha una storia così lunga come quella della crittografia a chiave simmetrica, essendo stata scoperta solo a metà degli anni '70; di conseguenza non ci sono molte informazioni circa l'effettiva affidabilità di questi metodi.

2. Knapsack Ciphers

2.1. Problema Knapsack (discreto)

Supponiamo di avere a disposizione un insieme di oggetti di peso diverso e supponiamo di prenderne un certo numero per infilarli in uno zaino di capacità nota; il problema consiste nel determinare quali sono gli oggetti da inserire in modo da non eccedere la capacità dello zaino ed, allo stesso tempo, riempito il più possibile.

In maniera più formale, dato un insieme (a_1, a_2, \dots, a_n) con $a_i \in \mathbb{N} - \{0\}$ ed $S \in \mathbb{N} - \{0\}$, il problema dello zaino richiede di trovare quali di questi numeri interi, se ce ne sono, possono essere sommati per raggiungere (al massimo) S . In definitiva, si chiede di **risolvere l'equazione**:

$$(1) \quad S = a_1x_1 + a_2x_2 + \dots + a_nx_n = \sum_{i=1}^n a_ix_i \quad \text{con} \quad \begin{cases} S = \text{capienza dello zaino} \\ a_i = \text{pesi dei singoli oggetti} \\ x_i = "0" \text{ oppure } "1" \quad \forall i = 1, 2, \dots, n \end{cases}$$

Esempio 1

Sia $(a_1, a_2, a_3, a_4, a_5) = (2, 7, 8, 11, 12)$ ed $S = 21$. Facendo qualche calcolo si vede come:

$$21 = 2 + 8 + 11$$

$$21 = 2 + 7 + 12$$

Conseguentemente, ci sono due soluzioni per il relativo problema $21 = 2x_1 + 7x_2 + 8x_3 + 11x_4 + 12x_5$:

$$\begin{cases} x_1 = x_3 = x_4 = 1 \\ x_2 = x_5 = 0 \end{cases}$$

$$\begin{cases} x_1 = x_2 = x_5 = 1 \\ x_3 = x_4 = 0 \end{cases}$$

2.1.1. Complessità

Per **verificare** che una singola configurazione sia soluzione per la (1) basta fare n addizioni; ma per **trovare** una soluzione (per tentativi) ci vuole un tempo pari alla verifica di tutte le 2^n possibili di combinazioni degli a_n coefficienti, tramite un algoritmo greedy. Il miglior metodo conosciuto per trovare una soluzione richiede $O(2^{n/2})$ bit operations, che comunque rende l'operazione non polinomiale e quindi computazionalmente complessa.

2.2. Problema Knapsack semplificato – Sequenze super-increasing

Certi valori di interi (a_1, a_2, \dots, a_n) rendono la soluzione del problema più semplice rispetto alla soluzione del caso generale:

1. se $a_j = 2^{j-1}$ allora risolvere l'equazione (1) richiede semplicemente di trovare lo sviluppo in binario del numero S ;
2. una **sequenza super-increasing** è un insieme (a_1, a_2, \dots, a_n) con $a_i \in \mathbb{N} - \{0\}$ scelti in modo che la somma dei primi $j-1$ sia sempre minore del j -esimo, ossia:

$$\sum_{i=1}^{j-1} a_i < a_j$$

Esempio 2

E' facile verificare che la sequenza $(2, 3, 7, 14, 27)$ è supercrescente.

Tramite l'algoritmo e l'esempio successivo si mostrerà che, se si sceglie una sequenza super-increasing, trovare una soluzione del problema risulta molto più semplice.

2.2.1. Algoritmo 1 – Soluzione con sequenze super-increasing

Per risolvere la (1), ossia il problema dello zaino, utilizzando una sequenza supercrescente (a_1, a_2, \dots, a_n) , basta utilizzare il seguente algoritmo scritto in pseudo-codice oppure la versione scritta in Pascal:

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. if $S \geq a_n$ then $x_n = 1$ 2. else $x_n = 0$ 3. if $S - \sum_{i=j+1}^n a_i x_i \geq a_j$ then $x_j = 1$ 4. else $x_j = 0$ | <ol style="list-style-type: none"> 1. for $i=1$ to n do 2. if $S \geq a_n$ then begin 3. $x_n = 1$; 4. $S = S - a_n$; 5. end; 6. else $x_n = 0$; 7. end; |
|---|---|

Figura 2 Pseudo-codice a sx e algoritmo scritto in Pascal a dx.

L'algoritmo non fa altro che procedere "a ritroso", verificando se è possibile selezionare i vari oggetti. Analizzando lo pseudo-codice, si parte (righe 1-2) dal più grande degli interi a_n e si verifica se questo è minore del limite massimo S (capienza dello zaino); se sì, si pone $x_n = 1$ (lo si mette nello zaino), altrimenti lo si scarta. A questo punto (righe 3-4) si procede con i restanti interi $x_{n-1}, x_{n-2}, \dots, x_1$ con la stessa metodologia.

Per verificare la **correttezza** dell'algoritmo si procede in questo modo. Visto che ogni x_i può assumere valore "0" oppure "1" (il che equivale a prendere o meno l'oggetto), se alla riga 1 si

ponesse $x_n = 0$ si avrebbe che $\sum_{i=1}^n a_i x_i \leq \sum_{i=1}^{n-1} a_i < a_n \leq S$, contraddicendo la condizione secondo cui

$\sum_{j=1}^n a_j x_j = S$. In maniera analoga, se alla riga 3 si fosse posto $x_j = 0$ si sarebbe avuto che

$$\sum_{i=1}^n a_i x_i \leq \sum_{i=1}^{j-1} a_i + \sum_{i=j+1}^n a_i x_i < a_j + \sum_{i=j+1}^n a_i x_i \leq S.$$

Esempio 3

Presa la sequenza super-increasing $(2,3,7,14,27)$ e ponendo $S=37$, la soluzione dell'equazione (1) si trova come segue:

$$S = 37 \geq 27 \Rightarrow \begin{cases} x_5 = 1 \\ 37 - 27 = 10 \end{cases} \Rightarrow 10 = 2x_1 + 3x_2 + 7x_3 + 14x_4$$

$$10 < 14 \Rightarrow x_4 = 0 \Rightarrow 10 = 2x_1 + 3x_2 + 7x_3$$

$$7 \geq 2 + 3 \Rightarrow \begin{cases} x_3 = 1 \\ 10 - 7 = 3 \end{cases} \Rightarrow 3 = 2x_1 + 3x_2$$

$$3 \geq 3 \Rightarrow \begin{cases} x_2 = 1 \\ 3 - 3 = 0 \end{cases} \Rightarrow x_1 = 0$$

Quindi la soluzione è data da:

$$\begin{cases} x_2 = x_3 = x_5 = 1 \\ x_1 = x_4 = 0 \end{cases} \Rightarrow 37 = 3 + 7 + 27$$

2.3. Knapsack Ciphers

Il sistema crittografico a chiave pubblica, basato sul problema NP-completo del knapsack, fu sviluppato da **Merkle** e **Hellman** nel 1978. C'è da dire che, come qualsiasi algoritmo di crittografia, all'aumentare della lunghezza della chiave aumenta proporzionalmente la complessità per un crittoanalista che vuole forzare il sistema. Malgrado questo, però, **non è stato mai implementato nella pratica** in quanto, in alcune circostanze, l'algoritmo risulta vulnerabile (indipendentemente dalla lunghezza delle chiavi).

2.3.1. Cifratura – Premesse

La fase di cifratura si basa su una sequenza supercrescente modificata, in modo da rendere mascherata tale sequenza e quindi “difficile” la decifrazione a chi non è l'effettivo destinatario del messaggio. In particolare, sia (a_1, a_2, \dots, a_n) una sequenza super-increasing, sia $m \in \mathbb{N} - \{0\}$ con $m > 2a_n$ e sia $w \in \mathbb{Z} : mcd(m, w) = 1$ con inverso \bar{w} modulo m . Si crea la **sequenza modificata** (b_1, b_2, \dots, b_n) dove:

$$(2) \quad b_j \equiv wa_j \pmod{m} \quad \text{con } 0 \leq b_j < m$$

Tale sequenza non può essere utilizzata per risolvere il problema (1) tramite l'Algoritmo 1, visto che **non è più super-increasing**. Tuttavia, se \bar{w} è noto, si può calcolare:

$$\bar{w}S = \sum_{i=1}^n \bar{w}b_i x_i \equiv \sum_{i=1}^n a_i x_i \pmod{m} \quad \text{visto che } \bar{w}b_j \equiv a_j \pmod{m}$$

Si ha quindi che il problema originale è dato da:

$$(3) \quad S_0 = \sum_{i=1}^n a_i x_i \quad \text{con } S_0 = \text{resto positivo minimo di } \bar{w}S \pmod{m}$$

e quest'ultima equazione è facile da risolvere perché, a questo punto, (a_1, a_2, \dots, a_n) è una sequenza supercrescente. Per riportarsi al problema modificato basta osservare che $b_j \equiv wa_j \pmod{m}$ e $0 \leq b_j < m$.

Esempio 4

Preso la sequenza supercrescente $a = (3, 5, 9, 20, 44)$, questa può essere facilmente trasformata nella sequenza $b = (23, 68, 69, 5, 11)$ calcolando $b_j \equiv wa_j \pmod{m}$ con $w = 67$, $m = 89$. Quindi, per risolvere il problema dello zaino $S = 84 = 23x_1 + 68x_2 + 69x_3 + 5x_4 + 11x_5$ basta moltiplicare ambo i membri dell'equazione per $\bar{w} = 4$ (inverso di $w = 67 \pmod{89}$) e successivamente ridurre il tutto modulo 89, per ottenere la congruenza $3x_1 + 5x_2 + 9x_3 + 20x_4 + 44x_5 \equiv 84 \cdot 4 = 336 \equiv 69 \pmod{89}$. A questo punto si è riportato il problema originale S a quello più semplice S_0 , nel quale si utilizzano sequenze super-increasing, e si può quindi procedere alla sua risoluzione tramite l'Algoritmo 1:

$$\begin{cases} x_2 = x_4 = x_5 = 1 \\ x_1 = x_3 = 0 \end{cases}$$

Quindi il problema originale S ha soluzione $84 = 68 + 5 + 11$.

2.3.2. Cifratura – Algoritmo

Il sistema di crittografia sviluppato da Merkle e Hellman funziona come segue; il destinatario Bob sceglie individualmente:

1. una sequenza supercrescente (a_1, a_2, \dots, a_n) con $a_i \in \mathbb{N} - \{0\}$ di lunghezza n
2. un modulo $m \in \mathbb{N} - \{0\}$ con $m > 2a_n$
3. un $w \in \mathbb{Z} : mcd(m, w) = 1$ con inverso \bar{w} modulo m

e rende pubblica la sequenza modificata (b_1, b_2, \dots, b_n) , che rappresenta la **chiave pubblica** di cifratura e . Quando Alice vuole mandare un messaggio P a Bob, per prima cosa trasforma il messaggio in chiaro P in una sequenza di "0" ed "1" secondo la Tabella 1. Successivamente,

essendo questo in cifrario a blocchi, la stringa binaria che rappresenta il messaggio in chiaro viene divisa in blocchi di lunghezza n (dove per semplicità si presuppone che la lunghezza della stringa binaria sia divisibile per n , altrimenti si completa con “1”). Per ogni blocco, si calcola la somma usando la sequenza modificata (b_1, b_2, \dots, b_n) , e quindi il blocco $x_1 x_2 \dots x_n$ permette di ottenere:

$$(4) \quad S = b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

Infine, le somme generate da ogni blocco formano il testo cifrato C da spedire a Bob.

<i>Lettera</i>	<i>Traduzione in binario</i>	<i>Lettera</i>	<i>Traduzione in binario</i>
A	00000	N	01101
B	00001	O	01110
C	00010	P	01111
D	00011	Q	10000
E	00100	R	10001
F	00101	S	10010
G	00110	T	10011
H	00111	U	10100
I	01000	V	10101
J	01001	W	10110
K	01010	X	10111
L	01011	Y	11000
M	01100	Z	11001

Tabella 1 Tabella di traduzione da lettere a binario.

2.3.3. Decifrazione – Premesse

Prima di tutto notiamo come sia difficile risalire al testo in chiaro P , partendo da quello cifrato C senza conoscere la **chiave di decifrazione** d (che nel nostro caso è la coppia (m, w)). In questo caso è infatti necessario risolvere un gruppo di problemi dello zaino della forma (4).

Se invece si conosce la coppia (m, w) , si può effettuare facilmente la procedura di decifrazione, in modo che Bob possa leggere il testo mandato da Alice. Infatti, il problema nella forma (4) può essere trasformato nel problema più semplice tramite:

$$\bar{w}S = \bar{w}b_1 x_1 + \bar{w}b_2 x_2 + \dots + \bar{w}b_n x_n = \sum_{i=1}^n \bar{w}b_i x_i \equiv a_1 x_1 + a_2 x_2 + \dots + a_n x_n = \sum_{i=1}^n a_i x_i \pmod{m}$$

ricordando che $\bar{w}b_j \equiv a_j \pmod{m}$ e \bar{w} è l'inverso di w modulo m , e si ha quindi la (3). Da notare è che entrambi i membri della (3) sono interi positivi minori di m (visto che sono congrui modulo m).

Esempio 5

Cifratura: il destinatario del messaggio sceglie:

1. una sequenza super-increasing $a_{10} = (2, 11, 14, 29, 58, 119, 241, 480, 959, 1917)$ con $n=10$
2. un modulo $m=3837$ con $m > 2a_{10}$

$$3. \text{ un } w = 1001 : mcd(m, w) = 1$$

La sequenza modificata b_{10} viene calcolata a partire dalla supercrescente a_{10} tramite la (2):

$$\begin{aligned} a_1 = 2 &\Rightarrow 1001 \cdot 2 \pmod{3837} = 2002 = b_1 & a_2 = 11 &\Rightarrow 1001 \cdot 11 \pmod{3837} = 3337 = b_2 \\ a_3 = 14 &\Rightarrow 1001 \cdot 14 \pmod{3837} = 2503 = b_3 & a_4 = 29 &\Rightarrow 1001 \cdot 29 \pmod{3837} = 2170 = b_4 \\ a_5 = 58 &\Rightarrow 1001 \cdot 58 \pmod{3837} = 503 = b_5 & a_6 = 119 &\Rightarrow 1001 \cdot 119 \pmod{3837} = 172 = b_6 \\ a_7 = 241 &\Rightarrow 1001 \cdot 241 \pmod{3837} = 3347 = b_7 & a_8 = 480 &\Rightarrow 1001 \cdot 480 \pmod{3837} = 855 = b_8 \\ a_9 = 959 &\Rightarrow 1001 \cdot 959 \pmod{3837} = 709 = b_9 & a_{10} = 1917 &\Rightarrow 1001 \cdot 1917 \pmod{3837} = 471 = b_{10} \end{aligned}$$

e si ha quindi che $b_{10} = (2002, 3337, 2503, 2170, 503, 172, 3347, 855, 709, 471)$; questa è la chiave di cifratura e viene resa pubblica, in modo da essere utilizzata da chiunque voglia mandare un messaggio cifrato ad destinatario.

Volendo spedire il messaggio $P = \text{FUNZIONA}$, il primo passo dell'algoritmo prevede la trasformazione in binario del testo e successivamente il raggruppamento in blocchi da $n=10$ bits:

$$\underbrace{\overbrace{00101}^F + \overbrace{10100}^U}}_{\text{blocco 1}} + \underbrace{\overbrace{01101}^N + \overbrace{11001}^Z}}_{\text{blocco 2}} + \underbrace{\overbrace{01000}^I} + \underbrace{\overbrace{01110}^O}}_{\text{blocco 3}} + \underbrace{\overbrace{01101}^N} + \underbrace{\overbrace{00000}^A}}_{\text{blocco 4}}$$

A questo punto, per ogni blocco di $n=10$ bits, il mittente calcola la somma dei corrispondenti termini della sequenza modificata/pubblica b_{10} :

$$\text{Blocco 1}(FU) \rightarrow 00101 \ 10100 = 2503 + 503 + 172 + 855 = 4033$$

$$\text{Blocco 2}(NZ) \rightarrow 01101 \ 11001 = 3337 + 2503 + 503 + 172 + 3347 + 471 = 10279$$

$$\text{Blocco 3}(IO) \rightarrow 01000 \ 01110 = 3337 + 3347 + 855 + 709 = 8248$$

$$\text{Blocco 4}(NA) \rightarrow 01101 \ 00000 = 3337 + 2503 + 503 = 6343$$

Infine, il mittente spedisce la sequenza di numeri appena calcolata (che rappresenta il testo cifrato), che nel nostro caso è $C = (4033, 10279, 8248, 6343)$.

Decifrazione: il destinatario riceve $C = (4033, 10279, 8248, 6343)$ e calcola:

$$1. \text{ l'inverso di } w \pmod{m} = 1001 \pmod{3837} \rightarrow \bar{w} = 23$$

$$2. \ S_0 \text{ tramite la (3)}$$

e quindi, per risolvere il problema Knapsack semplificato, si utilizza la sequenza super-increasing $a_{10} = (2 = x_1, 11 = x_2, 14 = x_3, 29 = x_4, 58 = x_5, 119 = x_6, 241 = x_7, 480 = x_8, 959 = x_9, 1917 = x_{10})$ scelta precedentemente e tenuta segreta. In particolare, per decifrare il vettore C si calcola:

$$\text{Blocco 1} \rightarrow \begin{cases} 23 \cdot 4033 \equiv 671 \pmod{3837} \\ 671 = 480 + 119 + 58 + 14 \Rightarrow x_3 + x_5 + x_6 + x_8 \Rightarrow 00101\ 10100 \Rightarrow FU \end{cases}$$

$$\text{Blocco 2} \rightarrow \begin{cases} 23 \cdot 10279 \equiv 2360 \pmod{3837} \\ 2360 = 1917 + 241 + 119 + 58 + 14 + 11 \Rightarrow x_2 + x_3 + x_5 + x_6 + x_7 + x_{10} \Rightarrow 01101\ 11001 \Rightarrow NZ \end{cases}$$

$$\text{Blocco 3} \rightarrow \begin{cases} 23 \cdot 8248 \equiv 1691 \pmod{3837} \\ 1691 = 959 + 480 + 241 + 11 \Rightarrow x_2 + x_7 + x_8 + x_9 \Rightarrow 01000\ 01110 \Rightarrow IO \end{cases}$$

$$\text{Blocco 4} \rightarrow \begin{cases} 23 \cdot 6343 \equiv 83 \pmod{3837} \\ 83 = 58 + 14 + 11 \Rightarrow x_2 + x_3 + x_5 \Rightarrow 01101\ 00000 \Rightarrow NA \end{cases}$$

Infine, basterà concatenare le coppie di lettere decifrate da ogni blocco per ottenere il testo originale P =FUNZIONA.

2.4. Implementazione pratica

Un uso reale del sistema prevede uno zaino di circa 200 oggetti ed il valore di ogni termine della sequenza supercrescente ha una lunghezza compresa tra 200 e 400 bits; tali valori sono prodotti da un generatore di sequenze casuali.

Usando un algoritmo brute-force, un computer che provi un milione di possibilità al secondo, troverebbe la soluzione in 10^{46} anni; un milione di queste macchine, pur lavorando in parallelo, risolverebbero il problema in un tempo comunque non accettabile.

2.5. Difetti del Knapsack Ciphers

Il cifrario basato sul problema del knapsack sembrava essere un eccellente candidato per essere utilizzato in sistemi di sicurezza basati su chiavi pubbliche. Tuttavia, nel 1982 Shamir [15] trovò un metodo efficiente per risolvere la sequenza modificata (b_1, b_2, \dots, b_n) , dove ogni b_i segue la (2), che impiega “soltanto” $O(n)$ bit operations (quindi polinomiale, più tosto che esponenziale). Altri dettagli sono esposti in [9]. Infatti, non ci sono voluti milioni di computer per rompere il sistema, ma soltanto un buon crittoanalista. Per primo Herlestan notò che spesso un singolo bit del testo in chiaro poteva essere recuperato; allora Shamir mostrò che il sistema poteva essere rotto in certe circostanze. Dopo vari tentativi, Shamir e Zippel trovarono un punto debole che permetteva di ricostruire la sequenza super-increasing dalla sequenza modificata tramite l'utilizzo di un algoritmo chiamato L^3 .

Sono state presentate molte varianti del sistema, ma quasi tutte sono state rotte. L'unica che resiste è la variante di Chor-Rivest.

3. Bibliografia

- 1) R. C. Merkle, M. E. Hellman,
"Hiding Information and Signatures in Trapdoor Knapsacks",
IEEE Transactions on Information Theory, Vol. 24, No. 5, pp. 525-530, September 1978.
- 2) [Definition of Signature scheme] A. Shamir,
"A Fast Signature Scheme",
MIT/LCS/TM-107, MIT Laboratory for Computer Science, July 1978.
- 3) M. E. Hellman,
"The Mathematics of Public-Key Cryptography",
Scientific American, Vol. 241, No. 8, pp. 146-157, August 1979.
- 4) G.J. Simmons (editor),
"Section 10.2 Knapsack Cryptosystems and Section 10.3 Generalized Knapsack Cryptosystems",
Contemporary Cryptography, IEEE Press, 1992.
- 5) B. Schneier,
"Section 19.2 Knapsack Algorithms",
[*Applied Cryptography, Second Edition*](#), John Wiley & Sons, 1996.
- 6) A.J. Menezes, P.C. v.Oorschot, S.A. Vanstone,
"Section 8.6 Knapsack Public-Key Encryption",
[*Handbook of Applied Cryptography*](#), CRC Press, 1996.
- 7) D.R. Stinson,
"Section 5.3 The Merkle-Hellman Knapsack System",
[*Cryptography : Theory and Practice*](#), CRC Press, 1995.
- 8) P. Garrett,
"Section 10.6 Knapsack Ciphers",
[*Making, Breaking Codes : An Introduction to Cryptology*](#), Prentice-Hall, 2000.
- 9) A. M. Odlyzko,
"The Rise and Fall of Knapsack Cryptosystems",
Cryptography and Computational Number Theory, Am. Math. Soc., Proc. Symp. Appl. Math., Vol. 42, pp. 75-88, 1990.
<http://www.research.att.com/~amo/doc/crypto.html>
- 10) E.R. Berlekamp, R.J. McEliece, H.C.A. van Tilborg,
"On the Inherent Intractability of Certain Coding Problems",
IEEE Transactions on Information Theory, Vol. IT-24, pp. 384-386, 1978.
- 11) G. Brassard,
"A Note on the Complexity of Cryptography",
IEEE Transactions on Information Theory, Vol. IT-25, No. 5, pp. 232-233, 1979.
- 12) B. Chor, R.L. Rivest,
"A Knapsack Type Cryptosystem Based on Arithmetic in Finite Fields",
Advances in Cryptology - CRYPTO '84 Proceedings, pp. 54-65, Plenum Press, 1985.
IEEE Transactions on Information Theory, Vol. IT-34, No. 5, pp. 901-909, 1988.
<http://theory.lcs.mit.edu/~rivest/publications.html>
- 13) S. Vaudenay,
"Cryptanalysis of the Chor-Rivest Cryptosystem",
Advances in Cryptology - CRYPTO '98 Proceedings, pp. 243-256, Springer-Verlag, 1998.
<http://www.dmi.ens.fr/~vaudenay/pub.html#Vau98h>

- 14) B. Przydatek,
"A Fast Approximation Algorithm for the Subset-Sum Problem",
Invited presentation at IFORS'99, 1999.
<http://www.inf.ethz.ch/departement/TI/um/research/publications/>
- 15) A. Shamir,
"A Polynomial-Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem",
Advances in Cryptology - CRYPTO '82 Proceedings, pp. 279-288, Plenum Press, 1983.
IEEE Transactions on Information Theory, Vol. IT-30, pp. 699-704, 1984.
- 16) A. M. Odlyzko,
"Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Signature Scheme",
IEEE Transactions on Information Theory, Vol. IT-30, pp. 594-601, 1984.
<http://www.research.att.com/~amo/doc/crypto.html>
- 17) E.F. Brickell, J.C. Lagarias, A. M. Odlyzko,
"Evaluation of the Adleman Attack on Multiply Iterated Knapsack Cryptosystems",
Advances in Cryptology - CRYPTO '83 Proceedings, pp. 39-42, Plenum Press, 1984.
<http://www.research.att.com/~amo/doc/crypto.html>