Tecniche di Progettazione Automatica **Anno Acc. 1995-96**
PROF. ALFREDO DE SANTIS

SISTEMI CRITTOGRAFICI A CHIAVE PUBBLICA

a cura di Sergio Arnese, Gaetano Cuomo e Tiziana De Rosa *9 maggio 1996*

Contents

1 Introduzione

Nei sistemi crittografici a chiave privata la sicurezza della cifratura dipende o dalla sicurezza della procedura di cifratura e decifratura oppure dalla segretezza del metodo con cui viene cifrato il messaggio. Ovviamente questo porta a degli inconvenienti del tipo:

1. Prima che una qualsiasi informazione possa essere cifrata su un canale non sicuro, mittente e destinatario devono convenire sulla chiave. Dato che la sicurezza del sistema dipende esclusivamente dalla segretezza della chiave, questa deve essere trasmessa mediante un canale sicuro, per esempio un corriere fidato che è un mezzo lento e costoso.
2. La distribuzione delle chiavi diventa un problema serio quando le persone che desiderano la privatezza non si siano mai scambiate alcuna comunicazione, o quando si debba mantenere la privatezza su una rete estesa (situazioni piuttosto frequenti nelle relazioni commerciali).
3. L'autentificazione dei messaggi, infatti un sistema crittografico tradizionale non è in grado di assicurare in modo completo l'autentificazione di un messaggio. Dato che mittente e destinatario condividono la stessa chiave, non c'è modo di impedire che il destinatario invii messaggi a se stesso che poi sembrino provenire dal mittente.

Per superare tali ostacoli fu introdotto da Whitfield Diffie e Martin Hellman nel 1976 il concetto di crittografia a chiave pubblica. Le innovazioni relative a questo nuovo concetto furono presentate da Diffie ed Hellman nello stesso anno alla National Computer Conference e pochi mesi dopo apparve un articolo intitolato "New directions in Criptography" sull' IEEE Transaction on Information Theory [2].
I sistemi a chiave pubblica, chiamati anche sistemi a due chiavi o asimmetrici, differiscono dai sistemi convenzionali in quanto non c'é piú una singola chiave segreta condivisa da una coppia di utenti. Piuttosto, ogni utente ha una sua propria chiave materiale K; inoltre, la chiave materiale di ogni utente é divisa in due porzioni, una componente privata d_k ed una pubblica e_k. La componente pubblica genera una trasformazione pubblica $e_k(x)$ e la componente privata genera una trasformazione privata $d_k(x)$. In analogia al caso convenzionale, $e_k(x) e d_k(x)$ potrebbero essere definite, rispettivamente, funzioni di cifratura e decifratura.
In tale sistema possiamo avere

- $(d_k(e_k(M))) = M$
- $(e_k(d_k(M))) = M$

o entrambe le relazioni.
Uno dei requisiti di queste funzioni é che e_k deve essere una funzione **"one-way trapdoor"**. Una funzione é detta "one-way" quando risulta facile da calcolare ma difficile da invertire, a meno che non si é in possesso di una informazione segreta aggiuntiva -trapdoor-. Cosí nel nostro caso se (e_k é una funzione di questo tipo, la decifrazione da parte del crittoanalista risulta difficile da realizzare, mentre la stessa operazione risulta facile da parte di chi riceve il messaggio che é in possesso della componente privata (d_k che rappresenta la trapdoor.
Sfortunatamente però pur essendoci molte funzioni ritenute one-way, non esiste una teoria

matematica che permetta di dimostrare che una funzione gode di questa proprietá.
Un esempio di funzione ritenuta one-way é:

$$f(x) = x^a \bmod n$$

dove n è il prodotto di due primi p e q abbastanza grandi, a è un intero positivo ed $f : \mathbf{Z}_n \to \mathbf{Z}_n$.

1.1 Sicurezza e autenticitá dei sitemi a chiave pubblica

Per garantire la sicurezza, le trasformazioni di un sistema a chiave pubblica devono soddisfare $(d_k(e_k(M))) = M$. Supponendo che A voglia spedire un messaggio sicuro M a B. Allora :

- A ottiene la chiave pubblica di B, e_{kB} :
- A calcola il testo cifrato $C = e_{kB}(M)$ corrispondente al messaggio M e lo spedisce a B;
- B riceve il testo cifrato C e adopera la sua chiave privata per ottenere il messaggio M, $d_{kB}(C) = d_{kB}(e_{kB}(M)) = M$.

Se una trasmissione da A viene intercettata da un nemico, sicuramente non puó essere decifrata perché d_{kB} é privata, per questo motivo la sicurezza é garantita. Comunque, presumibilmente tutti possono accedere ad e_{kB}; B non ha modo di conoscere l'identitá del mittente, per cui una trasmissione di A potrebbe essere alterata. Quindi autenticitá e integritá non sono assicurate.
Per garantire autenticitá ed integritá le trasmissioni di un sistema a chiave pubblica devono soddisfare $d_k(e_k(M)) = M$.
Supponiamo che A voglia mandare un messaggio autentico a B. Allora:

- A calcola la sua firma digitale $C' = d_{kA}(M')$:
- A spedisce la coppia (M', C') (messaggio,firma);
- Quando B riceve la coppia (M', C') puó verificare che questo proviene da A essendo e_{kA} la chiave pubblica di A, verificando l'equazione $M' = e_{kA}(C')$.

Così B puó accertare che il messaggio é stato spedito da A e non é stato alterato. Infatti assumendo che M' é un testo in chiaro valido, B sa che C é stato spedito da A e non é stato alterato durante la trasmissione. Questo segue dalla natura "one-way" della funzione e_{kA}.
Se un crittoanalista parte da un messaggio M' potrebbe trovare C' tale che $e_{kA}(C') = M'$, questo implicherebbe che e_{kA} é invertibile il che é un assurdo.
Un'importante osservazione sui sistemi a chiave pubblica è che essi non forniscono una sicurezza assoluta. Infatti un nemico osservando il testo cifrato y, potrebbe codificare tutti i possibili testi in chiaro X usando la chiave pubblica e_K fino ad ottenere l'unico X tale che $y = e_K(X)$. Proprio per questo i sistemi crittografici a chiave pubblica vengono definiti concettualmente in termini di funzioni chiamate one-way trapdoor che rendono il problema computazionalmente difficile.

Nella seguente figura é rappresentato lo schema generale di un crittosistema a chiave pubblica:

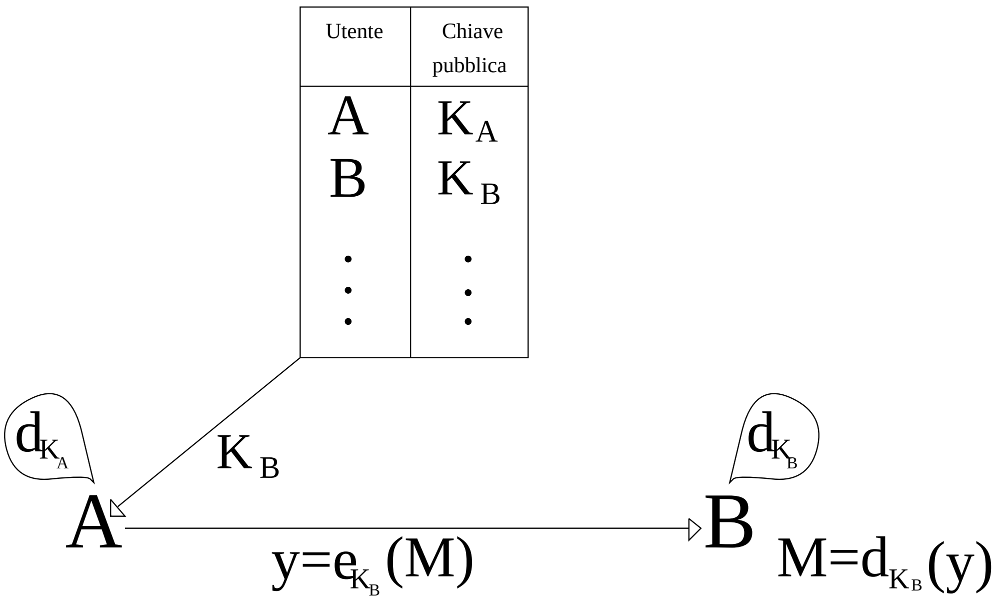

Figure 1: Schema generale di un crittosistema a chiave pubblica

1.2 Principali sistemi a chiave pubblica

Alcuni dei sistemi crittografici a chiave pubblica più conosciuti sono :

RSA [4] [5] La sicurezza dell'RSA si basa sulla difficoltà di fattorizzare numeri molto grandi.

RABIN La sicurezza del crittosistema di Rabin è anche essa basata sulla difficoltà di fattorizzare numeri molto grandi.

ELGAMAL Questo crittosistema basa la sua sicurezza sulla difficoltà del problema del calcolo del logaritmo discreto su campi finiti.

MERKLE-HELLMAN [5] Questo sistema è basato sulla difficoltà del problema del *subset sum* che è NP-completo; comunque tutte le variazioni del sistema dello zaino si sono dimostrate abbastanza insicure ad eccezione del sistema di Chor-Rivest.

Mc ELIECE Questo sistema è basato sulla teoria algebrica dei codici ed è considerato sicuro. Esso é costruito sul problema della decodificazione di un codice lineare che è NP-completo.

2 RSA

La prima realizzazione di un sistema crittografico a chiave pubblica fú proposta da Ron Rivest, Adi Shamir e Leonard Adleman nel 1977 e conosciuto con il nome di RSA.
Il crittosistema RSA é basato su un problema NP che consiste nel fattorizzare un numero molto grande (nell'ordine di 100 - 200 digits). Il problema della fattorizzazione dei numeri grandi veniva giá studiato nell'antica Grecia, e tuttora non esistono algoritmi efficienti per risolverlo.

Esempio 1 *Fattorizzare il numero 29083. Procedendo a mano ci impiegheremmo almeno un ora per trovare i fattori 127 e 229, mentre per verificare che si tratta dei fattori corretti basta qualche minuto.*

Proprio la difficoltà del problema suggerisce un punto di partenza per costruire una buona funzione one-way trapdoor. Comunque costruire una tale funzione non è un compito facile, ma questo ostacolo è stato brillantemente superato dagli autori del crittosistema RSA.

2.1 Sistema Crittografico RSA

Il crittosistema a chiave pubblica RSA si basa sul fatto che sebbene sia semplice sotto il profilo computazionale trovare dei numeri primi elevati, la scomposizione del prodotto di due numeri siffatti è a tutt'oggi computazionalmente irrealizzabile. L' RSA è costituito da una chiave pubblica e_K (nota a tutti) ed una chiave privata d_K (segreta). Formalmente si ha :

Definizione 1 *Sia $n = p \cdot q$, dove p e q sono interi primi molto grandi. Sia $\mathcal{P} = \mathcal{C} = \mathbf{Z}_n$, definiamo*

$$\mathcal{K} = \{(n, p, q, e, d) : n = pq,\ p \text{ e } q \text{ sono primi},\ ed \equiv 1 \bmod \varphi(n)\}$$

dove $\varphi(n) = (p-1)(q-1)$. Per $K = (n, p, q, e, d)$ definiamo :

$$e_K(x) = x^e \bmod n$$

e

$$d_K(y) = y^d \bmod n$$

con $x, y \in \mathbf{Z}_n$.

Per generare la chiave pubblica di cifratura, ogni utente del sistema RSA sceglie a caso due numeri primi piuttosto grandi p e q. Il prodotto $n = p \cdot q$ di questi due numeri primi, unito ad un numero e costituiscono la chiave pubblica (e, n) del sistema. Il numero e è scelto in modo che $\gcd(e, \varphi(n)) = 1$ dove $\varphi(n) = (p-1)(q-1)$ è la funzione di Eulero nota nella teoria dei numeri e definita come il numero di interi tra 1 ed n che non hanno fattori in comune con n. La chiave privata è costituita dalla coppia (d, n) dove d è calcolato come l'inverso moltiplicativo di $e \bmod \varphi(n)$; le proprietà di $\varphi(n)$ garantiscono sempre l'esistenza dell'inverso moltiplicativo d.
Per effettuare la cifratura di un messaggio $M \in \mathbf{Z}_n$ il mittente preleva dall'elenco la chiave pubblica dell'utente destinatario e calcola il testo cifrato

$$C = e_K(M) = M^e \bmod n.$$

Il destinatario conoscendo la sua chiave segreta (d, n) calcola il messaggio in chiaro

$$M = d_K(C) = C^d \bmod n.$$

Verifichiamo ora che la fase di cifratura e quella di decifratura sono una l'inversa dell'altra. Poichè $ed \equiv 1 \bmod \varphi(n)$ si ha $ed = t\varphi(n) + 1$ con t intero e $t \geq 1$. Supposto che $M \in \mathbf{Z}_n$ si ha :

$$C^d = M^{e^d} = M^{ed} = M^{t\varphi(n)+1} \bmod n = M$$

in quanto per il teorema di Eulero si ha che:

$$x^{\varphi(n)} = 1 \bmod n.$$

1. Bob genera due numeri primi grandi, p e q
2. Bob calcola $n = p * q$ e $\varphi(n) = (p-1)(q-1)$
3. Bob sceglie un numero $0 < e < \varphi(n)$ tale che $gcd(e, \varphi(n)) = 1$
4. Bob calcola $d_k = e_k^{-1}\, mod \varphi(n)$ usando l'algoritmo di Euclide
5. Bob pubblica n ed e in una directory come sua chiave pubblica

Esempio 2 *Nel presente esempio illustriamo lo schema a blocchi relativo alle azioni eseguite da due utenti che vogliono scambiarsi un messaggio utilizzando l' RSA:*

a = 00	b = 01	c = 02	d = 03	e = 04	f = 05	g = 06
h = 07	i = 08	j = 09	k = 10	l = 11	m = 12	n = 13
o = 14	p = 15	q = 16	r = 17	s = 18	t = 19	u = 20
v = 21	w = 22	x = 23	y = 24	z = 25	A = 26	B = 27
C = 28	D = 29	E = 30	F = 31	G = 32	H = 33	I = 34
J = 35	K = 36	L = 37	M = 38	N = 39	O = 40	P = 41
Q = 42	R = 43	S = 44	T = 45	U = 46	V = 47	W = 48
X = 49	Y = 50	Z = 51	0 = 52	1 = 53	2 = 54	3 = 55
4 = 56	5 = 57	6 = 58	7 = 59	8 = 60	9 = 61	= 62
. = 63	, = 64	; = 65	? = 66	...		

- Tabella 1 -

Nel sistema RSA ogni utente si procura due numeri casuali grandi p e q che gli servono da parametri segreti e un numero casuale e (1). L'utente rende noti come chiave di cifratura pubblica e ed n (2). Per cifrare il testo in chiaro P il mittente lo converte in una sequenza di numeri mediante l'alfabeto indicato nella tabella 1 (3), e quindi fraziona questa sequenza in blocchi di uguale lunghezza $P_1, P_2, \ldots,$ *in modo che ogni* P_i *sia minore di n (4). Ogni blocco viene trasformato in un numero del testo cifrato elevandolo alla* $e-$*esima potenza e quindi riducendolo modulo n (5). I numeri* $C_1, C_2, \ldots$ *del testo cifrato vengono trasmessi su un canale insicuro. Il destinatario sfrutta la propria chiave segreta per decifrare il messaggio ricevuto (6). Per trasformare* $C_1, C_2, \ldots$ *nei numeri del testo in chiaro, il destinatario eleva ognuno di essi alla* $d-$*esima potenza e lo riduce modulo n (7).*

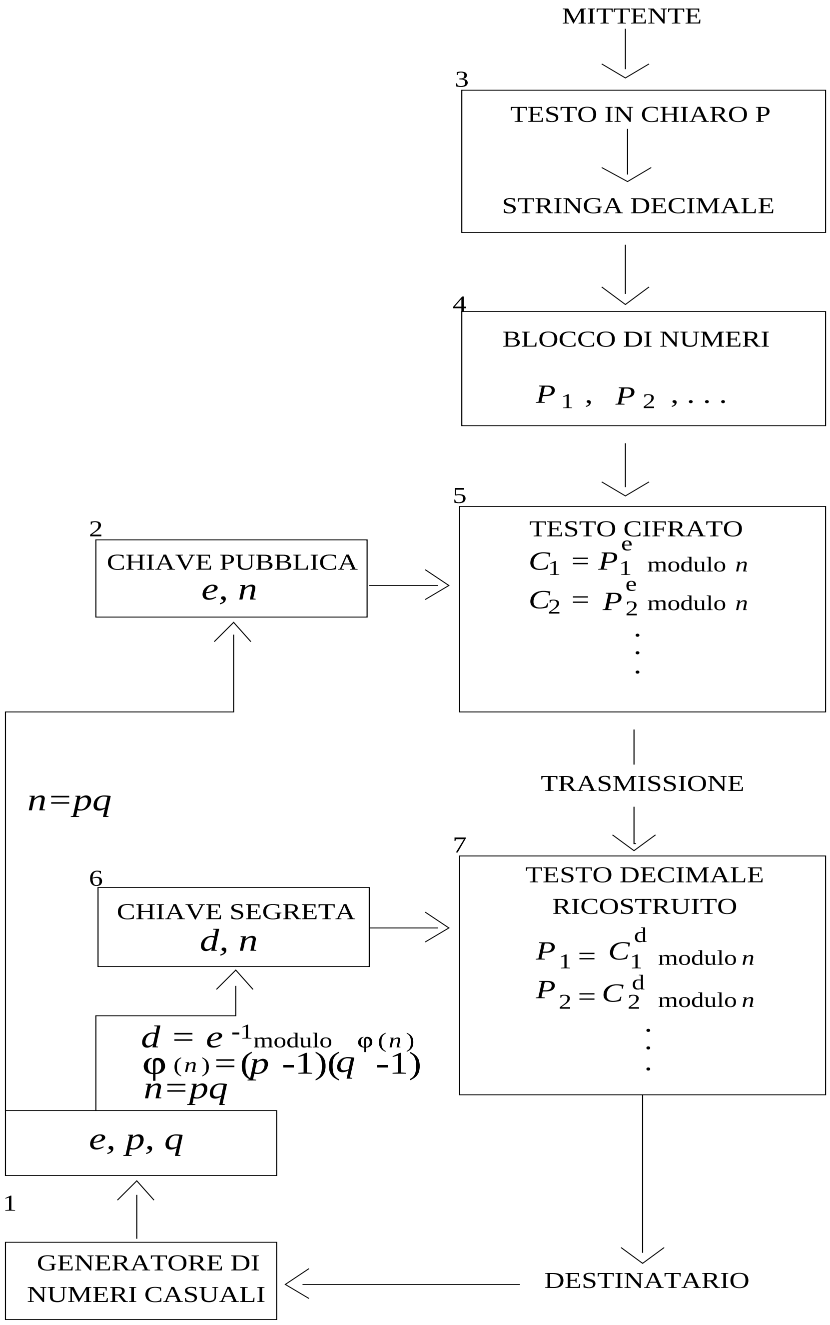

Figure 2: Schema generale

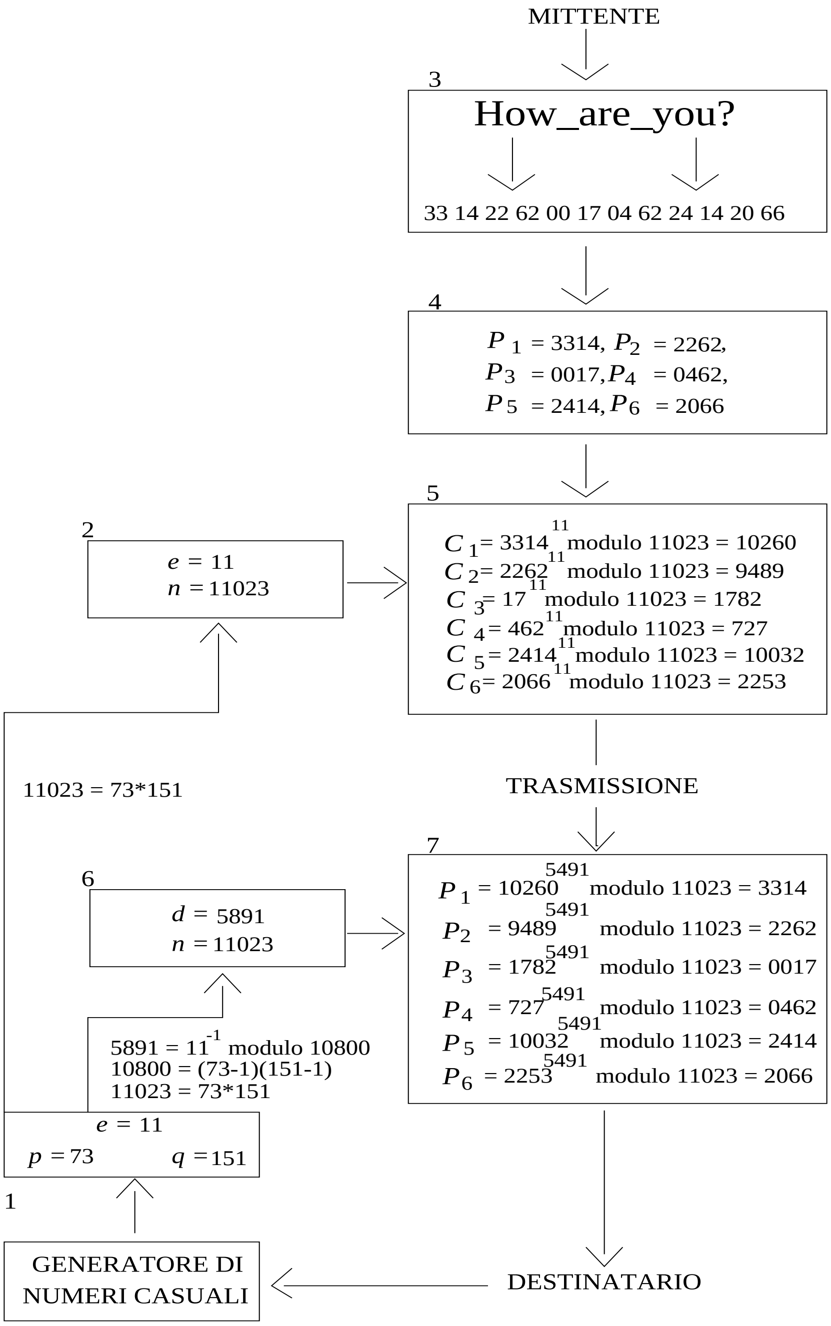

Figure 3: Esempio numerico

Esempio 3 *Supponiamo che Bob scelga* $p = 73$ *e* $q = 151$*. Allora* $n = p \cdot q = 11023$ *e* $\varphi(n) = (p-1)(q-1) = 72 \cdot 150 = 1080$*. Quindi* $1080 = 2^4 \cdot 3^3 \cdot 5^2$*, l'intero e usato nella fase di cifratura é tale da non essere divisibile per* $2, 3, 5$ *(in pratica Bob verifica che* $gcd(e, \varphi(n)) = 1$ *utilizzando l'algoritmo di Euclide). Supponiamo che Bob scelga* $e = 11$*, l'algoritmo esteso di Euclide calcola* $d = e^{-1} = 11^{-1} \bmod 10800$*; questa sarà l'informazione segreta di Bob. Bob pubblica* $n = 11023ed$ $e = 11$*. Ora supponiamo che Alice voglia spedire il messaggio "How_are_you?" a Bob. Facendo riferimento alla Tabella 1, Alice fa corrispondere al messaggio la sequenza di numeri:"33 14 22 62 00 17 04 62 24 14 20 66". Questa stringa viene divisa in* 6 *blocchi di* 4 *cifre ciascuno e ognuno di questi viene cifrato secondo la regola*

$C_i = (P_i)^e \bmod n$. *Alice allora spedisce i blocchi cifrati a Bob, il quale mediante la sua chiave pubblica ottiene il messaggio chiave mediante la trasformazione* $P_i = (C_i^d) \bmod n$.

2.2 Sicurezza dell' RSA

La sicurezza dell' RSA si basa sul fatto che la funzione di cifratura $e_K(x) = x^e \bmod n$ è una funzione one-way trapdoor che è computazionalmente difficile da invertire per un nemico che volesse decifrare un messaggio. Il **trapdoor** che viene utilizzato è **la conoscenza della fattorizzazione di** n. Infatti conoscendone la fattorizzazione é possibile calcolare

$$\varphi(n) = (p-1)(q-1)$$

e calcolare

$$d = e^{-1} \bmod \varphi(n)$$

usando l'algoritmo esteso di Euclide. Da ciò si deduce che la sicurezza dell' RSA dipende dal problema di fattorizzare grandi numeri. Di conseguenza il modo piú ovvio di attaccare il sistema, é quello di tentare di risolvere quest'ultimo problema. Gli algoritmi piú efficienti, conosciuti fino ad ora, hanno una complessitá $O(e^{\sqrt{(lnn)(lnlnn)}})$.
Attualmente numeri con 110 cifre decimali sono fattorizzabili, quindi per sicurezza si scelgono valori di p e q in modo che n abbia un numero di cifre superiore a 110. Alcune implementazioni hardware dell'RSA usano $512 - bit$ (circa 154 cifre) per n, mentre in realizzazioni software si arriva anche a $664 - bit$ (circa 200 cifre); per una maggiore sicurezza si arrivano ad usare anche $1024 - bit$ (circa 308 cifre). Fattorizzare un numero di $664 - bit$ richiede almeno 10^{23} passi usando gli algoritmi piú efficienti di cui sopra; per cui ipotizzando di avere una rete costituita da un milione di computers con ciascuno di essi che esegue un milione di passi al secondo, il tempo impiegato per fattorizzare n sarebbe dell'ordine dei 4000 anni. Se poi n fosse un numero a $1024 - bit$, la stessa rete impiegherebbe 10^{10} anni per fattorizzarlo.
Ció sottolinea l'osservazione fatta nell'introduzione, e cioé che non é impossibile decifrare un testo cifrato con il crittosistema RSA, ma piuttosto é computazionalmente difficile.
Notiamo esplicitamente che non é stata dimostrata l'equivalenza tra la sicurezza dell'RSA e la fattorizzazione di numeri grandi o problemi simili. Con ció si intende dire che specifici protocolli possono essere rotti, come si avrá modo di vedere negli attacchi considerati piú avanti.

2.3 Crittoanalisi dell'RSA

Osserviamo che per un crittoanalista rompere l'RSA equivale a calcolare $\varphi(n)$. Infatti, se n e $\varphi(n)$ sono conosciuti ed n è il prodotto di due primi p e q, n può essere facilmente fattorizzato risolvendo le due seguenti equazioni:

$$n = p \cdot q$$

$$\varphi(n) = (p-1)(q-1)$$

nelle due incognite q e p. Sostituendo $q = n/p$ nella seconda equazione, si ottiene un' unica equazione di secondo grado nella sola incognita p :

$$p^2 - (n - \varphi(n) + 1)p + n = 0$$

Le due radici di questa equazione sono i fattori p e q. Quindi se un crittoanalista conosce il valore di $\varphi(n)$ può fattorizzare n e rompere il sistema.

Esempio 4 *Supponiamo che il crittoanalista conosca* $\varphi(n) = 84754668$ *e* $n = 84773093$. *Queste informazioni gli permettono di scrivere l'equazione:*

$$p^2 - 18426p + 84773093 = 0$$

Risolvendo l'equazione si ottengono le due radici 9539 e 8887, che sono i fattori p *e* q *di* n.

2.3.1 Individuazione dell'esponente di decifrazione

Proviamo ora che qualsiasi algoritmo che calcoli l'esponente di decrittazione d, può essere usato come un **Oracolo** in un algoritmo probabilistico che fattorizza n efficientemente. Dimostreremo che calcolare d è tanto difficile quanto fattorizzare n. L'algoritmo che andiamo a descrivere è un algoritmo probabilistico di tipo **Las Vegas**.

Definizione 2 *Sia* $0 \leq \varepsilon < 1$ *un numero reale, un algoritmo Las Vegas è un algoritmo probabilistico tale che per ogni istanza* I *del problema, può fallire nel dare una risposta con probabilità* ε. *Comunque se l'algoritmo ritorna una risposta allora la risposta deve essere corretta.*

Nota Un algoritmo *Las Vegas* può non dare una risposta, ma qualsiasi risposta esso dia questa è corretta; a differenza di un algoritmo *Montecarlo* il quale dà comunque una risposta, corretta o sbagliata che sia.

Se usiamo un algoritmo *Las Vegas* per risolvere un problema non facciamo altro che eseguire l'algoritmo più volte finchè non dà una risposta. La probabilità che non dia una risposta in m esecuzioni successive è ε^m; in media il numero di volte che l'algoritmo deve essere eseguito per ottenere una risposta è $1/\varepsilon$.

Supponiamo di avere un' algoritmo **A** che calcoli l'esponente di decrittazione d dato quello di crittazione e (che è pubblico). Vediamo come un algoritmo *Las Vegas* usa **A** come un *Oracolo* fattorizzando n con probabilità almeno $1/2$. Quindi se eseguiamo l'algoritmo m volte, n sarà fattorizzato con probabilità almeno $1 - \frac{1}{2^m}$. L'algoritmo si basa sulle radici quadrate di $1 \bmod n$, dove $n = pq$ è il prodotto di due primi distinti e dispari. La congruenza $x^2 \equiv 1 \bmod p$ ha due soluzioni, $x \equiv \pm 1 \bmod p$. Ciò deriva dal fatto che $p \mid (x-1)(x+1)$ e che il primo p divide o $(x-1)$ oppure $(x+1)$. Analogamente la congruenza $x^2 \equiv 1 \bmod q$ ha due soluzioni, $x \equiv \pm 1 \bmod q$. Quindi $x^2 \equiv 1 \bmod n \Leftrightarrow x^2 \equiv 1 \bmod p$ e $x^2 \equiv 1 \bmod q$; da questo segue che $x^2 \equiv 1 \bmod n \Leftrightarrow x \equiv \pm 1 \bmod p$ e $x \equiv \pm 1 \bmod q$. Allora ci sono quattro radici quadrate di $1 \bmod n$ che possono essere trovate usando il *teorema Cinese del resto.* Due di queste soluzioni sono $x \equiv \pm 1 \bmod n$ che sono dette banali; le altre due dette non banali sono le opposte delle prime modulo n. Supponiamo che x sia una radice non banale di $1 \bmod n$, allora si ha:

$$n \mid (x-1)(x+1)$$

ma n non divide nessuno dei due fattori; da ciò segue che il $\gcd((x+1), n)$ è uguale a p oppure a q; analogamente il $\gcd((x-1), n)$ è uguale a p oppure a q. Il gcd può essere calcolato usando l'algoritmo di Euclide senza conoscere la fattorizzazione di n. Dunque la conoscenza di una radice quadrata non banale di $1 \bmod n$ produce la fattorizzazione di n in tempo polinomiale. Di seguito presentiamo un algoritmo che, facendo uso di **A** come un *Oracolo*, fattorizza n trovando una radice non banale di $1 \bmod n$.

1. Scegliere a caso w tale che $1 \leq w \leq n-1$
2. Calcolare $x = \gcd(w, n)$
3. **if** $1 < x < n$ **then quit** (successo $x = p$ oppure $x = q$)
4. Calcolare $d = \mathbf{A}(e)$
5. Scrivi $de - 1 = 2^s r$, con r dispari
6. Calcola $v = w^r \bmod n$
7. **if** $v \equiv 1 \bmod n$ **then quit** (fallimento)
8. **while** $v \not\equiv 1 \bmod n$ **do**
9. $\quad v_0 = v$
10. $\quad v = v^2 \bmod n$
11. **if** $v_0 \equiv -1 \bmod n$ **then quit** (fallimento)
12. $\qquad$ **else** $x = \gcd((v_0 + 1), n)$ (successo $x = p$ oppure $x = q$)

Esempio 5 *Siano* $n = 89855713$, $e = 34986517$, $d = 82330933$, *e il valore scelto casualmente per* w *è 5. Si ha :*

$$de - 1 = 2^3 \cdot 360059073378795$$

Nel passo 6. $v = 85877701$ *e nel passo* 10. $v = 1$*; nel passo* 12. *calcoliamo gcd(*$85877702, n$*) che è 9103. Questo è uno dei fattori di* n*, l'altro si ricava dalla semplice operazione* $\frac{n}{9103} = 9871$.

Passiamo ora all'analisi dell'algoritmo. Per prima cosa osserviamo che se siamo abbastanza fortunati nello scegliere w come multiplo di p o di q, possiamo fattorizzare n immediatamente (passi 2. e 3.). Se w è relativamente primo con n, allora calcoliamo $w^r, w^{2r}, w^{4r}, \ldots$ da successivi quadrati, finchè $w^{2^t r} \equiv 1 \bmod n$ per alcuni t. Dato che $de - 1 = 2^s r \equiv 0 \bmod \varphi(n)$, sappiamo che $w^{2^s r} \equiv 1 \bmod n$. Dunque il ciclo while termina dopo al più s iterazioni; alla fine del ciclo troviamo un valore v_0 tale che $v_0^2 \equiv 1 \bmod n$, ma $v_0 \not\equiv 1 \bmod n$. Se $v_0 \equiv -1 \bmod n$ allora l'algoritmo fallisce altrimenti v_0 è una radice non banale di $1 \bmod n$, riuscendo cosí a fattorizzare n (passo 12.).

Dimostriamo ora che l'algoritmo ha successo con probabilità almeno 1/2. Ci sono due modi in cui l'algoritmo può fallire la fattorizzazione di n:

1. $w^r \equiv 1 \bmod n$ (passo 7.)
2. $w^{2^t r} \equiv -1 \bmod n$ per $0 \leq t \leq s-1$ (passo 11.)

Si devono considerare $s+1$ congruenze; se un valore casuale w è soluzione di almeno una delle $s+1$ congruenze, questa è una **cattiva** scelta e l'algoritmo fallisce. Quindi procediamo contando il numero di soluzioni di ciascuna di queste congruenze. Per prima si considera la congruenza $w^r \equiv 1 \bmod n$; il modo di analizzare una congruenza come questa è considerare le soluzioni mod p e mod q separatamente e combinarle poi utilizzando il *teorema Cinese del resto*. Notiamo che $x \equiv 1 \bmod n \Leftrightarrow x \equiv 1 \bmod p$ e $x \equiv 1 \bmod q$. Pertanto prima consideriamo $w^r \equiv 1 \bmod p$. Dato che p è un numero primo, risulta che $\mathbf{Z}_p^\star$ è un gruppo ciclico. Sia g un generatore mod p, possiamo scrivere $w = g^u$ per un unico intero $0 \leq u \leq (p-2)$. Allora si ha:

$$w^r \equiv 1 \bmod p$$

$$g^{ur} \equiv 1 \bmod p$$

$$(p-1) \mid (ur)$$

Consideriamo

$$(p-1) = 2^i p_1$$

dove p_1 è dispari, e

$$(q-1) = 2^j q_1$$

dove q_1 è dispari. Poichè

$$\varphi(n) = (p-1)(q-1) \mid (de-1) = 2^s r$$

abbiamo che

$$2^{i+j} q_1 p_1 \mid 2^s r$$

dunque

$$(i+j) \leq s$$

e

$$p_1 q_1 \mid r$$

Ora la condizione $(p-1) \mid (ur)$ diventa $2^i p_1 \mid (ur)$; poichè $p_1 \mid r$ ed r è dispari, è necessario e sufficiente che $2^i \mid u$. Pertanto $u = k2^i$ con $0 \leq k \leq p_1 - 1$, e il numero di soluzioni della congruenza $w^r \equiv 1 \bmod p$ è p_1.

Attraverso un'analoga discussione si giunge al risultato che il numero di soluzioni della congruenza $w^r \equiv 1 \bmod q$ è q_1. Possiamo combinare le soluzioni modulo p con le soluzioni modulo q per ottenere un'unica soluzione modulo n, utilizzando il *teorema Cinese del resto*. Di conseguenza il numero di soluzioni della congruenza $w^r \equiv 1 \bmod n$ è $p_1 q_1$. Il passo successivo è considerare una congruenza del tipo $w^{2^t r} \equiv -1 \bmod n$ per un fissato valore $0 \leq t \leq s-1$. Analogamente a quanto fatto sopra, analizziamo le congruenze modulo p e modulo q (si noti che $w^{2^t r} \equiv -1 \bmod n \Leftrightarrow w^{2^t r} \equiv -1 \bmod p$ e $w^{2^t r} \equiv -1 \bmod q$). Consideriamo per prima $w^{2^t r} \equiv -1 \bmod p$. Definiamo $w = g^u$; analogamente a prima abbiamo :

$$g^{u 2^t r} \equiv -1 \bmod p$$

dato che $g^{\frac{p-1}{2}} \equiv -1 \bmod p$, abbiamo che

$$u 2^t r \equiv \frac{p-1}{2} \bmod (p-1)$$

$$(p-1) \mid \left(u 2^t r - \frac{p-1}{2} \right)$$

$$2(p-1) \mid (u 2^{t+1} r - (p-1))$$

poichè $p - 1 = 2^i p_1$ abbiamo

$$2^{i+1} p_1 \mid (u 2^{t+1} r - 2^i p_1)$$

ponendo p_1 in evidenza otteniamo

$$2^{i+1} \mid \left(\frac{u2^{t+1}r}{p_1} - 2^i\right)$$

ora se $t \geq i$, allora possono non esserci soluzioni, poichè $2^{i+1} \mid 2^{t+1}$, ma $2^{i+1} \not| \, 2^i$. Viceversa se $t \leq (i-1)$, allora u è una soluzione se e solo se u è un dispari multiplo di 2^{i-t-1} (si noti che r/p_1 è un intero dispari).

Il numero di soluzioni in questo caso è

$$\frac{p-1}{2^{i-t-1}}\frac{1}{2} = 2^t p_1$$

Con un ragionamento simile si dimostra che la congruenza $w^{2^t r} \equiv -1 \bmod q$ non ha soluzioni se $t \geq j$ ed ha $2^t q_1$ soluzioni se $t \leq (j-1)$. Utilizzando il *teorema Cinese del resto*, si ha che il numero di soluzioni della congruenza $w^{2^t r} \equiv -1 \bmod n$ è

$$\begin{array}{rl} 0 \text{ se } & t \geq min\{i,j\} \\ 2^{2t} p_1 q_1 \text{ se } & t \leq min\{i,j\} - 1. \end{array}$$

Ora t può essere compreso tra 0 ed $(s-1)$. Senza perdere di generalità supposto $i \leq j$, il numero di soluzioni è 0 se $t \geq i$. Il numero totale di scelte **cattive** per w è al più:

$$p_1q_1 + p_1q_1(1 + 2^2 + 2^4 + \cdots + 2^{2i-2}) = p_1q_1\left(1 + \frac{2^{2i}-1}{3}\right)$$

$$= p_1q_1\left(\frac{2}{3} + \frac{2^{2i}}{3}\right)$$

sapendo che $(p-1) = 2^i p_1$ e $(q-1) = 2^j q_1$. Essendo $j \geq i \geq 1$ si ha :

$$p_1q_1 < \frac{n}{4}$$

abbiamo anche che

$$2^{2i}p_1q_1 \leq 2^{i+j}p_1q_1 = (p-1)(q-1) < n$$

dunque otteniamo

$$p_1q_1\left(\frac{2}{3} + \frac{2^{2i}}{3}\right) < \frac{n}{6} + \frac{n}{3} = \frac{n}{2}$$

Pertanto al più $\frac{n-1}{2}$ scelte di w sono **cattive**. Ne segue che almeno $\frac{n-1}{2}$ scelte sono **buone**, per cui la probabilità di successo dell'algoritmo è almeno 1/2.

2.3.2 Attacco mediante conoscenza di informazioni parziali sui bit del testo in chiaro

La conoscenza parziale di informazioni riguardanti i bits del testo in chiaro del messaggio cifrato con l'RSA può essere utile per costruire un attacco al crittosistema.

Diamo alcune definizioni :

1. Dato $y = e_K(x)$, definiamo *parity(y)* come il bit meno significativo di x

2. Dato $y = e_K(x)$, definiamo *half(y)* come :

$$half(y) = 0 \ se \ 0 \leq x < \frac{n}{2}$$

$$half(y) = 1 \ se \ \frac{n}{2} < x \leq (n-1)$$

Da questo si può dimostrare che dato $y = e_K(x)$, un qualsiasi algoritmo che calcoli *parity(y)* oppure *half(y)* può essere usato come un *Oracolo* per costruire un algoritmo che decodifichi il testo cifrato. Comunque vedremo che dato un testo cifrato calcolare il bit meno significativo del testo in chiaro è polinomialmente equivalente a determinare l'intero testo. Proviamo per prima che calcolare *parity(y)* è polinomialmente equivalente a calcolare *half(y)*. Infatti questo segue dalla conoscenza delle seguenti identità:

$$half(y) = parity(ye_K(2) \bmod n)$$

$$parity(y) = half(ye_K(2^{-1}) \bmod n)$$

e dalla regola moltiplicativa:

$$e_K(x_1)e_K(x_2) = e_K(x_1x_2).$$

Diamo ora un algoritmo che calcola $x = d_K(y)$ usando un *Oracolo* che calcola *half(y)*

1. Poni $k = \lfloor log_2 n \rfloor$
2. **for** $i = 0$ **to** k **do**
3. $\quad y_i = half(y)$
4. $\quad y = (ye_K(2)) \bmod n$
5. $lo = 0$
6. $hi = n$
7. **for** $i = 0$ **to** k **do**
8. $\quad mid = \frac{hi+lo}{2}$
9. $\quad$ **if** $y_i = 1$ **then** $lo = mid$
10. $\quad\quad$ **else** $hi = mid$
11. $x = \lfloor hi \rfloor$

Nei passi da 2. a 4. viene calcolato $y_i = half(y(e_K(2))^i) = half(e_K(x2^i))$ per $0 \leq i \leq log_2 n$. Osserviamo che

$$\begin{aligned}
half(e_K(x)) &= 0 \Leftrightarrow x \in \left[0, \tfrac{n}{2}\right) \\
half(e_K(2x)) &= 0 \Leftrightarrow x \in \left[0, \tfrac{n}{4}\right) \cup \left[\tfrac{n}{2}, \tfrac{3}{4}n\right) \\
half(e_K(4x)) &= 0 \Leftrightarrow x \in \left[0, \tfrac{n}{8}\right) \cup \left[\tfrac{n}{4}, \tfrac{3}{8}n\right) \cup \left[\tfrac{n}{2}, \tfrac{5}{8}n\right) \cup \left[\tfrac{3}{4}n, \tfrac{7}{8}n\right)
\end{aligned}$$

e cosí via. Quindi possiamo trovare x con una tecnica di ricerca binaria fatta nei passi 7. - 11.

Esempio 6 *Sia $n = 1457$, $e = 779$ e il testo cifrato y è 722. $e_K(2) = 946$. Supponiamo di usare un Oracolo che calcoli half ottenendo i seguenti valori di y_i nel passo 3.:*

i	0	1	2	3	4	5	6	7	8	9	10
y_i	1	0	1	0	1	1	1	1	1	0	0

- Tabella 3 -

La ricerca binaria procede nel seguente modo:

i	l_0	mid	hi
0	0.00	728.50	1457.00
1	728.50	1092.75	1457.00
2	728.50	910.62	1092.75
3	910.62	1001.69	1092.75
4	910.62	956.16	1001.69
5	956.16	978.92	1001.69
6	978.92	990.30	1001.69
7	990.30	996.00	1001.69
8	996.00	998.84	1001.69
9	998.84	1000.26	1001.69
10	998.84	999.55	1000.26
	998.84	999.55	999.55

- Tabella 4 -

Dunque il testo in chiaro è $x = \lfloor 999.55 \rfloor = 999$

2.3.3 Attacco Chosen-Ciphertext

Mostriamo questo tipo di attacco nei seguenti casi :

CASO 1 : Consideriamo due utenti di una rete, Oscar e Alice, e supponiamo che Oscar intercetti le comunicazioni di Alice prelevando i messaggi cifrati C. Matematicamente Oscar vuol calcolare $M = C^d$. Per calcolare M egli sceglie per primo un numero casuale r tale che $r < n$, ed inoltre conosce la chiave pubblica e di Alice. Egli può calcolare :

$$x = r^e \bmod n$$
$$y = xC \bmod n$$
$$t = r^{-1} \bmod n$$

Se $x = r^e \bmod n$ allora $r = x^d \bmod n$ per cui $t = x^{-d} \bmod n$. Ora Oscar manda y ad Alice chiedendone la firma, e Alice spedisce ad Oscar $u = y^d \bmod n$ (Alice ha firmato y e non una sua versione hash). Ora Oscar può calcolare :

$$tu \bmod n = x^{-d}y^d \bmod n = x^{-d}x^dC^d \bmod n = C^d \bmod n = M$$

CASO 2 : Supponiamo di avere un computer Hall che notifichi documenti. In altre parole se un utente vuole farsi notificare un documento, lo spedisce ad Hall, il quale lo firma con la sua chiave privata, e lo rispedisce al mittente. (Non vengono utilizzate funzioni one-way hash). Supponiamo che Oscar voglia farsi notificare un documento N da Hall il quale per una ragione qualsiasi non voglia farlo. Per prima cosa Oscar sceglie un valore arbitrario X e calcola $Y = X^e$, prelevando la chiave pubblica e di Hall. In seguito calcola $M = YN$ e fa firmare M da Hall. Quindi Hall gli ritorna $M^d \bmod n$, da ciò Oscar calcola

$$(M^d \bmod n)X^{-1} = N^d \bmod n$$

cioè la firma del documento N. Questo è possibile in quanto l'esponenziazione preserva la struttura moltiplicativa dell'input, cioè :

$$(XM)^d = X^d M^d \bmod n$$

CASO 3 : Oscar vuole far firmare a Bob un documento M_3 che Bob non vuole firmare. Allora Oscar, genera due documenti M_1, M_2 tali che :

$$M_3 \equiv M_1 M_2 \bmod n$$

Quindi manda alla firma di Bob sia M_1 che M_2 e calcola

$$M_3^d \bmod n = M_1^d \bmod n M_2^d \bmod n$$

che è la firma di M_3.

2.3.4 Attacco Common Modulus

Supponiamo che due utenti usano un modulo comune n ed esponenti di crittazione e_1 ed e_2 con GCD(e_1,e_2)=1. Se viene inviato lo stesso messaggio M ad entrambi, allora vengono calcolati i testi cifrati come:

$$C_1 = M^{e_1} \bmod n$$

$$C_2 = M^{e_2} \bmod n$$

Se entrambi C_1, C_2 vengono intercettati allora un intruso puó, usando l'algoritmo di Euclide, calcolare r ed s tali che :

$$re_1 + se_2 = 1.$$

e quindi ottenere

$$M = C_1^r C_2^s \bmod n.$$

2.3.5 Attacco Low Exponent

A priori l'esponente di crittazione e nello schema RSA è arbitrario. Comunque viene suggerito [5] scegliere un valore di e in modo da essere lo stesso per tutti gli utenti del sistema. Scegliere $e = 2$ è impossibile in quanto 2 non è primo con $p - 1$ o $q - 1$. Ci sono comunque interessi teorici nella scelta del valore 2 in quanto Rabin e Williams hanno notato che modificando l' RSA ed usando 2 come esponente, un attacco del tipo ciphertext-only è tanto difficile quanto fattorizzare. La scelta del valore 3 può essere fatta nell' RSA; essa comporta vantaggi dovuti al semplicità della codifica e decodifica dei messaggi. Questo valore ed altri esponenti dello stesso ordine di grandezza sono consigliati anche se comunque soffrono di qualche problema. Illustriamo cosa comporta la scelta del valore $e = 3$. Per prima cosa gli utenti devono essere sicuri che ogni blocco M che deve essere codificato soddisfi la relazione $M^3 \gg n$. È consigliabile rispettare tale relazione perchè essendo $C = M^3 \bmod n$ diviene semplicemente $C = M^3$ se $M^3 < n$. In questo caso la ricerca di M si riduce al semplice problema del calcolare una radice cubica. Cosí l' uso di $e = 3$ fa si che essa possa essere scelta in un sottoinsieme di $[0, n)$ piuttosto che l'intero intervallo come sarabbe preferibile. Ci possono essere altri problemi nello scegliere $e = 3$. Infatti supponiamo che A mandi lo stesso messaggio ad ognuno dei B_i, con $i = 1, 2, 3$. Supponiamo che i vari B_i usino come modulo n_i e $M < n_i$ per ogni i. Assumiamo che ognuno degli $\{n_1, n_2, n_3\}$ siano generati come prodotto di due numeri primi casuali; la probabilità fra i sei primi usati è circa zero. Dunque può essere soddisfacente assumere che gli n_i siano coppie relativamente prime. Sia $C_i = M^3 \bmod n_i$. Supponiamo che i tre messaggi cifrati siano intercettati. Usando il *teorema Cinese del resto*, il nemico può trovare x nell' intervallo $[0, n')$, con $n' = n_1 n_2 n_3$ dove $x \equiv C_i \bmod n_i$ con $i = 1, 2, 3$. Quindi $x \equiv M \bmod n'$. Essendo $M^3 < n'$ risulta $M^3 = x$, cosí $M = x^{\frac{1}{3}}$. Quindi il testo in chiaro M può essere facilmente recuperato dai tre testi cifrati. Concludendo, l'uso di $e = 3$ o di altri esponenti dello stesso ordine di grandezza, rendono l' RSA vulnerabile rispetto ad attacchi di tipo ciphertext-only. Il mittente può cercare di modificare M per ogni destinatario in modo da evitare questo attacco. Hastad ha dimostrato che in generale la scelta di un basso esponente rende comunque l' RSA vulnerabile se ci sono dipendenze lineari tra parti del testo in chiaro.

2.4 RSA in Hardware

Sono stati costruiti molti chips che implementano a livello hardware l'RSA; nella seguente tabella ne vengono elencati alcuni:

marca	clock velocità	baud rate per 512 bits	clock cicli per 512 bit codifica	tecnologia	bits per chip	numero di transistors
Alpha Techn.	25 Mhz	13 K	0.98 M	2 micron	1024	180,000
AT&T	15 Mhz	19 K	0.4 M	1.5 micron	298	100,000
British Telecom.	10 Mhz	5.1 K	1 M	2.5 micron	256	–
Business Sim. Ltd.	5 Mhz	3.8 K	0.67 M	gate array	32	–
Calmos Syst. Inc.	20 Mhz	28 K	0.36 M	2 micron	593	95,000
Cnet	25 Mhz	5.3 K	2.3 M	1 micron	1024	100,000
Cryptech	14 Mhz	17 K	0.4 M	gate array	120	33,000
Cylink	16 Mhz	6.8 K	1.2 M	1.5 micron	1024	150,000
Pijnenburg	25 Mhz	50 K	0.256 M	1 micron	1024	400,000
Plessy Crypto.	–	10.2 K	–	–	512	–
Sandia	8 Mhz	10 K	0.4 M	2 micron	272	86,000

- Tabella 2 -

2.5 Velocità dell' RSA

L'inplementazione hardware più veloce dell' RSA è circa 1000 volte più lenta di quella del DES. Un' implementazione hardware con integrazione a larga scala (VLSI) abbastanza veloce con moduli a $512 - bit$ processa $64Kbps$. Attualmente sono disponibili chips che hanno una

capacità di $1 Mbps$ usando moduli a $512 - bit$.
Per quanto riguarda le implementazioni software, in generale il DES è circa 100 volte più veloce dell'RSA. Da quanto detto si deduce che nella pratica l'RSA viene usato solo per lo scambio di chiavi, mentre per cifrare l'intero messaggio viene usato un sistema crittografico più veloce come ad esempio il DES.

3 Crittosistema di Rabin

Anche il crittosistema di Rabin (1979) è un crittosistema a chiave pubblica. La rottura di tale sistema equivale alla fattorizzazione del modulo $n = pq$ per attacchi di tipo chosen-plaintext. Formalmente si ha:

Definizione 3 *Sia n il prodotto di due distinti primi p e q dove $p \equiv q \equiv 3 \bmod 4$.*
Siano $P = C = \mathbf{Z}_n$, e definiamo:

$$\mathcal{K} = \{(n, p, q, B) : n = pq, p \equiv q \equiv 3 \bmod 4,\ \ p \text{ e } q \text{ sono primi},\ \ 0 \leq B \leq n-1\}$$

Per $K = (n, p, q, B)$, definiamo :

$$e_K(x) = x(x + B) \bmod n$$

e

$$d_K(y) = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2}$$

I valori n e B sono pubblici, mentre p e q sono segreti.

Nel crittosistema di Rabin, la funzione di cifratura e_K non è iniettiva, cosí la fase di decifratura risulta essere ambigua. Infatti ci sono quattro possibili testi in chiaro che cifrati danno lo stesso testo cifrato.
Dato un messaggio in chiaro M, la codifica è data da:

$$C = e_K(M) = M(M + B) \bmod n$$

la decodifica

$$M = d_K(C) = \sqrt{\frac{B^2}{4} + C} - \frac{B}{2}$$

3.1 Fase di codifica

Vediamo ora più dettagliatamente la fase di codifica.
Sia ω una delle quattro radici di $1 \bmod n$, sia $M \in \mathbf{Z}_n$ un testo in chiaro, possiamo verificare la seguente relazione:

$$\begin{aligned} e_K\left(\omega\left(M + \tfrac{B}{2}\right) - \tfrac{B}{2}\right) &= \left(\omega\left(M + \tfrac{B}{2}\right) - \tfrac{B}{2}\right)\left(\omega\left(M + \tfrac{B}{2}\right) + \tfrac{B}{2}\right) \\ &= \omega^2\left(M + \tfrac{B}{2}\right)^2 - \tfrac{B^2}{4} \\ &= M^2 + MB + \tfrac{B^2}{4} - \tfrac{B^2}{4} \\ &= M^2 + MB \\ &= M(M + B) \\ &= e_K(M) \end{aligned}$$

I quattro possibili testi in chiaro che danno $e_K(M)$ sono $M, (-M-B), \omega\left(M+\frac{B}{2}\right)$, $-\omega\left(M+\frac{B}{2}\right)$, dove ω è una radice non banale di 1 mod n. In generale non esiste un modo che permetta di distinguere quale dei quattro possibili messaggi in chiaro sia quello giusto, a meno che il testo non contenga sufficiente ridondanza per eliminare tre di questi possibili valori.

3.2 Fase di decodifica

Passiamo ora all'analisi della fase di decodifica. Dato un testo cifrato y vogliamo determinare M tale che $M^2 + BM \equiv y \bmod n$. Questa è una equazione di secondo grado nell'incognita M; noi possiamo eliminare il termine lineare mediante la sostituzione $M_1 = M + \frac{B}{2}$ o equivalentemente $M = M_1 - \frac{B}{2}$. Da ciò si ha:

$$M_1^2 - BM_1 + \frac{B^2}{4} + BM_1 - \frac{B^2}{2} - y \equiv 0 \bmod n$$

oppure

$$M_1^2 \equiv \frac{B^2}{4} + y \bmod n$$

Se definiamo $C = \frac{B^2}{4} + y$, possiamo riscrivere la congruenza come:

$$M_1^2 \equiv C \bmod n$$

Da ciò la fase di decodifica si riduce all'estrazione di una radice quadrata modulo n; ciò è equivalente a risolvere le due congruenze:

$$M_1^2 \equiv C \bmod p$$

$$M_1^2 \equiv C \bmod q$$

Usando il *teorema Cinese del resto* si giunge alle quattro soluzioni combinando le due radici di C mod p con quelle di C mod q. Possiamo usare il criterio di Eulero per determinare se C è un *residuo quadratico* modulo p (e modulo q). Infatti C sarà un residuo quadratico modulo p e modulo q se la codifica è stata fatta correttamente. Comunque il criterio di Eulero non aiuta a trovare le radici di C; esso produce solo una risposta *"Si"* o *"No"*. Quando p è congruo a 3 mod 4, c'è una semplice formula per calcolare le radici di un residuo quadratico modulo p. Supponendo che C sia un residuo quadratico e p sia congruo a 3 mod 4 si ha:

$$(\pm C^{\frac{p+1}{4}})^2 = C^{\frac{p-1}{2}} C \equiv C \bmod p$$

Usando nuovamente il criterio di Eulero si ha che:
$C^{\frac{p-1}{2}} \equiv 1 \bmod p$ se C è un residuo quadratico modulo p. Dunque le due radici di C mod p sono $\pm C^{\frac{p+1}{4}}$; allo stesso modo le due radici di C mod q sono $\pm C^{\frac{q+1}{4}}$. In questo modo si ottengono le quattro radici quadrate di C mod n.

Nota Se $p \equiv 1 \bmod 4$ non esiste un algoritmo deterministico polinomiale che calcoli le radici di un residuo quadratico modulo p. Comunque esiste un algoritmo *Las Vegas* che lavora in tempo polinomiale.

Una volta determinati i quattro possibili valori di M_1 possiamo calcolare M dalla relazione:

$$M = M_1 - \frac{B}{2}$$

ottenendo i quattro possibili testi in chiaro.
Questo produce la formula di decodifica:

$$d_K(y) = \sqrt{\frac{B^2}{4} + y} - \frac{B}{2}$$

Esempio 7 *Illustriamo come funziona il crittosistema di Rabin. Sia* $n = 77 = 711$, *e* $B = 9$. *La funzione di cifratura è*

$$e_K(M) = M^2 + 9M \bmod 77$$

la funzione di decifratura è

$$d_K(y) = \sqrt{1 + y} - 43 \bmod 77$$

Supponiamo di voler decifrare il testo $y = 22$. *Per prima cosa è necessario trovare le radici di* $23 \bmod 7$ *e* $23 \bmod 11$. *Poichè 7 ed 11 sono entrambi congruenti a* $3 \bmod 4$ *si ha:*

$$23^{\frac{7+1}{4}} \equiv 2^2 \equiv 4 \bmod 7$$

$$23^{\frac{11+1}{4}} \equiv 1^3 \equiv 1 \bmod 11$$

Usando il teorema Cinese del resto *possiamo calcolare le quattro radici di* $23 \bmod 77$ *e cioè* $\pm 10, \pm 32 \bmod 77$. *I possibili quattro testi in chiaro sono:*

$$10 - 43 \equiv 44 \bmod 77$$

$$67 - 43 \equiv 24 \bmod 77$$

$$32 - 43 \equiv 66 \bmod 77$$

$$45 - 43 \equiv 2 \bmod 77$$

Si puó facilmente verificare che ognuno di questi quattro valori di testo in chiaro ottenuto, danno luogo allo stesso testo cifrato uguale a 22.

3.3 Sicurezza del crittosistema di Rabin

Proveremo che qualsiasi ipotetico algoritmo di decodifica **A**, può essere usato come un *Oracolo* in un algoritmo *Las Vegas* che fattorizzi il modulo n con probabilità almeno 1/2.

1. Scegliere un numero r a caso, con $1 \leq r \leq n - 1$
2. Calcola $y = r^2 - \frac{B^2}{4} \bmod n$
3. Calcola $x = \mathbf{A}(y)$
4. Calcola $x_1 = x + \frac{B}{2}$
5. **if** $x_1 \equiv \pm r \bmod n$ **then quit** (fallimento)
6. **else** $\gcd(x_1 + r, n) = p$ oppure q (successo)

Chiariamo alcuni punti:

- Osserviamo che $y = e_K\left(r - \frac{B}{2}\right)$ ritorna M nel passo 3.; infatti
$$e_K\left(r-\frac{B}{2}\right)=\left(r-\frac{B}{2}\right)^2+\left(r-\frac{B}{2}\right)B=r^2-rB+\frac{B^2}{4}+rB-\frac{B^2}{2}=r^2-\frac{B^2}{4}=y$$
nel passo 4. abbiamo $M_1^2 \equiv r^2 \bmod n$ da cui $M_1 \equiv \pm r \bmod n$ oppure $M_1 \equiv \pm\omega r \bmod n$.
- Abbiamo che $n \mid (M_1-r)(M_1+r)$ ma n non divide nessuno dei due fattori singolarmente. Dunque nel calcolo del $\gcd(M_1+r, n)$ (oppure $\gcd(M_1-r, n)$), deve produrre p o q e quindi la fattorizzazione di n.

3.4 Conclusioni

Il crittosistema di Rabin è sicuro contro un attacco chosen-plaintext. Comunque il sistema è completamente insicuro contro un attacco di tipo chosen-ciphertext. Infatti l'algoritmo mostrato sopra, usato per provare la sicurezza rispetto ad un attacco chosen-plaintext, può essere anche usato per rompere il sistema in un attacco chosen-ciphertext.

4 Crittosistema di ElGamal

Il crittosistema a chiave pubblica di ElGamal è basato sul problema del *logaritmo discreto*. Il problema del logaritmo discreto per campi finiti $\mathbf{Z}_p$ con p primo è cosí definito:

Definizione 4 *$I = (p, \alpha, \beta)$ dove p è primo, $\alpha \in \mathbf{Z}_p$ è un elemento primitivo e $\beta \in \mathbf{Z}_p^\star$ L'obiettivo è trovare un unico intero a con $0 \le a \le (p-2) : \alpha^a \equiv \beta \bmod p$. L'intero a sarà il $log_\alpha \beta$.*

In particolare non si conosce un algoritmo che lavori in tempo polinomiale per il problema del logaritmo discreto. Il motivo che porta ad utilizzare tale problema per costruire un sistema crittografico, stà nel fatto che questo è un problema difficile da risolvere; ma l'operazione inversa di esponenziazione può essere calcolata efficientemente. Definiamo formalmente il crittosistema di ElGamal.

Definizione 5 *Sia p un primo tale che il problema del logaritmo discreto in $\mathbf{Z}_p$ è intrattabile, e sia $\alpha \in \mathbf{Z}_p^\star$ un elemento primitivo. Sia $\mathcal{P} = \mathbf{Z}_p^\star$ e $\mathcal{C} = \mathbf{Z}_p^\star \times \mathbf{Z}_p^\star$ definiamo:*

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \bmod p\}$$

I valori p, α e β sono pubblici ed a è segreta. Per $K = (p, \alpha, a, \beta)$ e per un numero casuale segreto $k \in \mathbf{Z}_{p-1}$, definiamo:

$$e_K(M, k) = (y_1, y_2)$$

dove

$$y_1 = \alpha^k \bmod p$$

$$y_2 = M\beta^k \bmod p$$

Per y_1 e $y_2 \in \mathbf{Z}_p^\star$ definiamo:

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$$

Lo schema di ElGamal è uno schema non deterministico, quindi il testo cifrato dipende sia dal testo in chiaro M che dal valore k. Cosí possono esserci più testi cifrati che derivano da uno stesso testo in chiaro. In generale il sistema di ElGamal lavora in questo modo:
il testo in chiaro M è mascherato moltiplicandolo per β^k cosí da produrre y_2; il valore α^k è anch'esso trasmesso come parte del testo cifrato. Il destinatario conoscendo l'esponente segreto a, può calcolare β^k da α^k, e quindi rimuovere la mascheratura dividendo y_2 per β^k ottenendo M.

Esempio 8 *Supponiamo* $p = 2579, \alpha = 2, a = 765$ *e quindi* $\beta = 2^{765} \bmod 2579 = 949$. *Supponiamo ora che Alice voglia mandare il messaggio* $M = 1299$ *a Bob. Sia* $k = 853$*, Alice calcola:*

$$y_1 = 2^{853} \bmod 2579 = 435 \text{ e } y_2 = 1299 \cdot 949^{853} \bmod 2579 = 2396$$

Quando Bob riceve il testo cifrato $y = (435, 2396)$*, calcola*

$$M = 2396(435^{765})^{-1} \bmod 2579 = 1299$$

4.1 Equivalenza tra ElGamal e Diffie-Hellman

Dimostriamo ora che rompere il crittosistema di ElGamal è equivalente a risolvere il problema di DH. Supponiamo di avere un algoritmo **A** che risolve il problema DH e supponiamo di avere un messaggio (y_1, y_2) codificato con ElGamal. Applicando all'algoritmo **A** gli input p, α, y_1, β otteniamo:

$$\begin{aligned}\mathbf{A}(p, \alpha, y_1, \beta) &= \mathbf{A}(p, \alpha, \alpha^k, \alpha^a) \\ &= \alpha^{ka} \bmod p \\ &= \beta^k \bmod p\end{aligned}$$

Quindi la decodifica di (y_1, y_2) può essere facilmente calcolata come:

$$x = y_2(\beta^k)^{-1} \bmod p$$

Viceversa, supponiamo di avere un algoritmo **B** che rompe ElGamal; cioè dando in input a **B** i parametri $p, \alpha, \beta, y_1, y_2$ esso calcola la quantità

$$x = y_2(y_1^{log_\alpha \beta})^{-1} \bmod p$$

Dando ora in input a **B** i parametri p, α, β e γ del problema di DH, possiamo osservare che:

$$\begin{aligned}\mathbf{B}(p, \alpha, \beta, \gamma, 1)^{-1} &= 1((\gamma^{log_\alpha \beta})^{-1})^{-1} \bmod p \\ &= \gamma^{log_\alpha \beta} \bmod p\end{aligned}$$

5 Crittosistema di Merkle-Hellman

La prima descrizione di un crittosistema di tipo *knapsack*, fu fatta da Merkle ed Hellman nel 1978. Benchè questo sistema ed altre sue varianti sono stati rotti nel 1980, esso è ancora studiato per la sua eleganza concettuale e la sua fondamentale tecnica di progetto. Il crittosistema di Merkle-Hellman è basato sul problema del *subset sum*.

L'istanza del problema subset sum è:

Definizione 6 *$I = (s_1, \ldots, s_n, T)$ dove $s_1, \ldots, s_n$ e T sono interi positivi.*
Gli s_i sono i valori *e T è la* somma finale*.*
L'obiettivo è verificare se esiste un vettore binario

$$x = (x_1, \ldots, x_n) : \sum_{i=1}^{n} x_i s_i = T$$

Il problema del subset sum è un problema di decisione (cioé risponde solo "si" o "no") che può essere formulato come un problema di ottimizzazione quando si desidera trovare il vettore x tale che :

$$\sum_{i=1}^{n} x_i s_i = T.$$

Il problema del subset sum è NP-completo; cioè non è conosciuto un algoritmo che lo risolve in tempo poliniomiale nella sua forma generale, ma questo non esclude che ci sia un algoritmo che risolve particolari istanze in tempo polinomiale. Infatti definiamo un vettore *supercrescente* $(s_1, \ldots, s_n)$ se

$$s_j > \sum_{i=1}^{j-1} s_i \quad con \quad 2 \leq j \leq n$$

Se la lista dei valori è supercrescente allora il problema di ottimizzazione del subset sum può essere facilmente risolto in tempo lineare con n e se tale soluzione esiste é unica. Sia $s = (s_1, \ldots, s_n)$ una sequenza supercrescente, consideriamo la funzione:

$$e_s : \{0,1\}^n \rightarrow \left\{0, \ldots, \sum_{i=1}^{n} s_i\right\}$$

e definiamo la regola:

$$e_s(x_1 \ldots, x_n) = \sum_{i=1}^{n} x_i s_i.$$

Poichè s è una sequenza supercrescente ed e_s è iniettiva, un sistema crittografico basato su questa regola potrebbe essere rotto dallo stesso algoritmo che risolve il problema del subset sum in caso di sequenze supercrescenti. Comunque, un tale sistema potrebbe essere completamente insicuro dato che tutti possono decriptare un messaggio che é stato criptato in questo modo.
La strategia alternativa é perció trasformare la lista in modo tale che essa non é la piú lunga supercrescente mascherando tale sequenza. In fase di decodifica basterà applicare la trasformazione inversa per riottenere la sequenza supercrescente. Una trasformazione appropriata è la *trasformazione modulare.*
Si sceglie un primo $p : p > \sum_{i=1}^{n} s_i$ e si sceglie $a : 1 \leq a \leq (p-1)$.
Definiamo $t_i = a s_i \bmod p$ con $1 \leq i \leq n$. La lista dei valori $t = (t_1, \ldots, t_n)$ sarà la chiave pubblica, mentre i valori a e p usati per la trasformazione costituiranno la chiave segreta.
Naturalmente tale operazione puó essere ripetuta ottenendo versioni piú complesse; una prima iterazione si ottiene scegliendo un nuovo primo $p_1 > \sum_{i=1}^{n} t_i$ ed un valore $a_1 : 1 \leq a_1 \leq (p-1)$ tale che risulti $t_i' = a_1 t_i \bmod p_1$ con $1 \leq i \leq n$. Formalmente il crittosistema di Merkle-Hellman é definito in questo modo:

Definizione 7 *Sia $s = (s_1, \ldots, s_n)$ una sequenza supercrescente di interi, sia $p > \sum_{i=1}^{n} s_i$ un numero primo e sia $1 \leq a \leq (p-1)$.*

Per $1 \leq i \leq n$ *definiamo* $t_i = as_i \bmod p$ *e poniamo* $t = (t_1, \ldots, t_n)$.
Sia $\mathcal{P} = \{0,1\}^n, \mathcal{C} = \{0, \ldots, (p-1)^n\}$ *e sia*

$$\mathcal{K} = \{(s, p, a, t)\}$$

dove s, p, a, t *sono stati descritti precedentemente. Il valore* t *è pubblico ed i valori* p, a, s *sono segreti. Per* $K = (s, p, a, t)$ *definiamo*

$$e_K(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i t_i$$

Per $0 \leq y \leq (p-1)^n$, *definiamo* $z = a^{-1} y \bmod p$ *e risolvendo il problema del subset sum* $(s_1, \ldots, s_n, z)$ *si ottiene*

$$d_K(y) = (x_1, \ldots, x_n)$$

Esempio 9 *Sia* $s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$ *una sequenza supercrescente (la chiave segreta di Bob); sia* $p = 2003$ *ed* $a = 1289$. *Bob calcola la chiave pubblica* $t = (t_1, \ldots, t_9)$ *come:*

$$\begin{aligned}
t_1 &= 2 \cdot 1289 \bmod 2003 = 575 \\
t_2 &= 5 \cdot 1289 \bmod 2003 = 436 \\
t_3 &= 9 \cdot 1289 \bmod 2003 = 1586 \\
t_4 &= 21 \cdot 1289 \bmod 2003 = 1030 \\
t_5 &= 45 \cdot 1289 \bmod 2003 = 1921 \\
t_6 &= 103 \cdot 1289 \bmod 2003 = 569 \\
t_7 &= 215 \cdot 1289 \bmod 2003 = 721 \\
t_8 &= 450 \cdot 1289 \bmod 2003 = 1183 \\
t_9 &= 946 \cdot 1289 \bmod 2003 = 1570
\end{aligned}$$

Quindi la chiave pubblica sarà $t = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570)$.
Supponiamo ora che Alice voglia mandare a Bob il messaggio $x = 011000110101101110$, *essendo il messaggio più lungo della chiave pubblica di Bob, Alice divide* x *in blocchi di lunghezza pari a quella della chiave, quindi :*

$$x = \begin{cases} 011000110 \\ 101101110 \end{cases}$$

e calcola i valori che questi blocchi avranno, considerando che se nel blocco si presenta uno zero, il valore del corrispondente t_i *non verrà considerato. Infatti*

011000110	*corrisponde a*	$436 + 1586 + 721 + 1183 = 3926$
101101110	*corrisponde a*	$575 + 1586 + 1030 + 569 + 721 + 1183 = 5664$

Quindi Alice trasmetterà il testo cifrato $y = (3926, 5664)$.
Quando Bob riceve il messaggio cifrato y *calcola:*

$$\begin{aligned} z_1 &= a^{-1}y_1 \bmod p \\ &= 317 \cdot 3926 \bmod 2003 = 679 \end{aligned}$$

$$\begin{aligned} z_2 &= a^{-1}y_2 \bmod p \\ &= 317 \cdot 5664 \bmod 2003 = 800 \end{aligned}$$

Ottenuti questi valori, Bob ricava il messaggio in chiaro risolvendo il subset sum sulla sua sequenza supercrescente.

5.1 Implementazione pratica

Un uso reale del crittosistema di Merkle-Hellman prevede uno zaino di circa 200 oggetti, ed il valore di ogni termine della sequenza supercrescente ha una lunghezza compresa tra i 200 e i 400 bits.
Una reale implementazione usa un generatore di sequenze casuali per produrre tali valori. Usando un algoritmo di forza bruta, un computer che provi un milione di possibilità al secondo, troverebbe la soluzione in 10^{46} anni; un milione di macchine di questo tipo che lavorano in parallelo, risolverebbero il problema in un tempo comunque non accettabile.

5.2 Sicurezza dei crittosistemi knapsack

In realtà non ci sono volute un milione di macchine per rompere il sistema, ma un paio di buoni crittoanalisti. Per primo Herlestam notò che spesso un singolo bit del testo in chiaro poteva essere recuperato. Allora Shamir mostrò che il crittosistema poteva essere rotto in certe circostanze. Dopo vari tentativi falliti, Shamir e Zippel trovarono un punto debole che permetteva di ricostruire la sequenza supercrescente dalla sequenza normale usando un algoritmo chiamato L^3. Sono state presentate molte varianti di crittosistemi basati sul problema del subset sum, comprese le loro versioni iterate ma sono state tutte puntualmente rotte; l'unico che resiste è la variante di Chor-Rivest.

6 Crittosistema di Mc Eliece

Il crittosistema di Mc Eliece si basa su un problema NP-completo che è quello della *decodifica di un codice lineare a correzione di errore.*
In realtà la decodifica che viene trattata è un caso particolare del problema, mascherato in modo che sembri un istanza generale.
Diamo la seguente definizione:

Definizione 8 *Siano k ed n interi positivi tali che $k \leq n$. Un codice $C[n,k]$, è un sottospazio k-dimensionale di $\mathbf{Z}_2^n$, lo spazio vettoriale di tutte le n-ple binarie.*
Una **matrice generatrice** *per un $[n,k]$ codice C, è una matrice binaria $(k \times n)$ le cui righe costituiscono una base per C.*
Siano $x, y \in \mathbf{Z}_2^n$, dove $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$.
Definiamo la **distanza di Hamming***:*

$$d(x,y) = \ | \{i : 1 \leq i \leq n, x_i \neq y_i\}|$$

cioè il numero di coordinate in cui x e y differiscono.
Sia C un $[n,k]$ codice. Definiamo la distanza di C come :

$$d(C) = min\{d(x,y) : x, y \in C, x \neq y\}.$$

Un codice $[n,k]$ con distanza d viene denotato con un codice $[n,k,d]$.

Lo scopo di un codice a correzione di errore è di correggere gli errori casuali che occorrono nella trasmissione di dati su un canale rumoroso. Descriviamo questo brevemente.
Sia G una matrice generatrice per un codice $[n,k,d]$.
Supponiamo che x sia la k-pla binaria che vogliamo trasmettere. Allora Alice codifica x come la n-pla $y = xG$ e trasmette y sul canale.
Supponiamo che Bob riceva la n-pla r che può essere diversa da y. Egli decodifica r usando la seguente strategia (nearest neighbor decoding):

- Bob cerca la parola codice y' che è alla minima distanza da r. Allora decodifica r da y' determinando cosí la k-pla x' tale che $y' = x'G$.
- Bob spera che $y' = y$, cosí $x' = x$ (cioè tutti gli errori in trasmissione sono stati corretti).

È facile dimostrare che se il numero di errori occorsi è minore di $\frac{d-1}{2}$ questa strategia corregge tutti gli errori. Vediamo come funziona in pratica.
Se $|C| = 2^k$, e Bob deve comparare r con tutte le parole codice, dovrà esaminare 2^k vettori, che è un numero esponenzialmente grande rispetto a k. In altre parole, quest' algoritmo non lavora in tempo polinomiale.
Un altro approccio che é allla base di molti algoritmi di decodifica è basato sull'idea della **sindrome**. Una matrice *parity-check* per un $[n,k,d]$ codice C, genera una matrice binaria **G** di dimensioni $((n-k) \times n)$, che denotiamo con H, le cui righe formano una base per il complemento ortogonale di C, cioè $C^{\perp}$ è chiamato il codice **duale** di C.
In altre parole le righe di H sono vettori linearmente indipendenti e GH^T è una matrice di zeri di dimensioni $(k \times (n-k))$. Dato un vettore $r \in \mathbf{Z}_2^n$, definiamo la sindrome di r come Hr^T. Una sindrome è un vettore colonna con $(n-k)$ componenti.

Teorema 1 *Dato un codice $C[n,k]$, con matrice generatrice G e matrice parity-check H. Allora $x \in \mathbf{Z}_2^n$ è una parola codice se e solo se:*

$$Hx^T = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{pmatrix}$$

Inolte se $x \in C, e \in \mathbf{Z}_2^n$ ed $r = x + e$ allora $Hr^T = He^T$.

Pensiamo ad e come ad un vettore di errori che si verificano durante la trasmissione di una parola codice x. Allora r rappresenta il vettore ricevuto.
Il teorema 1 ci dice che la sindrome dipende solo dagli errori e non da particolari parole codice trasmesse. Questo suggerisce il seguente approccio di decodifica conosciuto come *sindrome*

decoding.
Inizialmente calcoliamo $\mathbf{s} = Hr^T$. Se $\mathbf{s}$ è un vettore di zeri allora la decodifica di r sarà r stessa. Se non è così si generano tutti i possibili vettori di errori di peso 1. Per ciascun vettore e calcoliamo He^T. Se $He^T = \mathbf{s}$ allora decodifichiamo r come $r - e$, altrimenti continuiamo a generare tutti i vettori di errori di peso $2, \ldots, \lfloor \frac{d-1}{2} \rfloor$. Appena si verifica che $He^T = \mathbf{s}$, allora decodifichiamo r come $r - e$.
Se ciò non si verifica mai si deduce che si sono verificati più di $\lfloor \frac{d-1}{2} \rfloor$ errori in trasmissione.
Con questo approccio possiamo decodificare un vettore ricevuto in al più

$$1 + \binom{n}{1} + \cdots + \binom{n}{\lfloor \frac{d-1}{2} \rfloor}$$

passi.
Questo metodo lavorerà su qualsiasi codice lineare.
Per certi specifici tipi di codici, la decodifica può essere velocizzata. Comunque la versione decisionale del problema del problema nearest neighbord decoding è NP-completo.
Un facile suggerimento per il crittosistema di Mc Eliece è di usare un codice di una classe di codici conosciuta come codici **Goppa**. Questi codici hanno infatti un efficiente algoritmo di decodifica. I parametri di questi codici hanno la forma $n = 2^m, d = 2t + 1$ e $k = n - mt$. Per una implementazione pratica di un crittosistema a chiave pubblica, Mc Eliece consiglia di prendere $m = 10$ e $t = 50$. Questo dà un codice Goppa che è un codice$[1024, 524, 101]$. Ogni testo in chiaro è un numero binario di 524-ple, e ogni testo cifrato è di 1024-ple. La chiave pubblica è una matrice binaria di (524×1024).
Formalmente il crittosistema di Mc Eliece é definito cosí.

Definizione 9 *Sia G una matrice generatrice per un codice Goppa $C[n, k, d,]$, dove $n = 2^m, d = 2t + 1$ e $k = n - mt$.*
Sia S una matrice $(k \times k)$ invertibile in $\mathbf{Z}_2$, sia P una matrice di permutazione $(n \times n)$ e sia $G' = S \cdot G \cdot P$. Sia $\mathcal{P} = \{0,1\}^k, \mathcal{C} = \{0,1\}^n$, e sia

$$\mathcal{K} = \{(G, S, P, G')\}$$

dove G, S, P e G' sono stati definiti precedentemente. G' è pubblico mentre G, S, P sono segreti.
Per $K = (G, S, P, G')$ definiamo:

$$e_K(x, e) = xG' + e$$

dove $e \in \mathbf{Z}_2^n$ è un vettore casuale di peso t.
Bob decodifica un testo cifrato $y \in \mathbf{Z}_2^n$ con le seguenti operazioni:

1. *Calcola $y_1 = yP^{-1}$*
2. *Decodifica y_1 ottenendo $y_1 = x_1 + e_1$ dove $x_1 \in C$*
3. *Calcola $x_0 \in \mathbf{Z}_2^k$ tale che $x_0 G = x_1$*
4. *Calcola $x = x_0 S^{-1}$*

Esempio 10 *La matrice*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

è una matrice generatrice per un codice $[7, 4, 3]$ *conosciuto come* codice di Hamming.
Supponiamo che Bob scelga le matrici

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

e

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Allora la matrice generatrice pubblica è:

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Ora supponiamo che Alice codifichi il testo $x = (1, 1, 0, 1)$ *usando un vettore di errori casuale di peso* 1 $e = (0, 0, 0, 0, 1, 0, 0)$.
Il testo cifrato calcolato come:

$$y = xG' + e = (1, 1, 0, 1) \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} + (0, 0, 0, 0, 1, 0, 0)$$

$$= (0, 1, 1, 0, 0, 1, 0) + (0, 0, 0, 0, 1, 0, 0) = (0, 1, 1, 0, 1, 1, 0)$$

Quando Bob riceve il testo cifrato y *per prima cosa calcola:*

$$y_1 = yP^{-1} = (0, 1, 1, 0, 1, 1, 0) \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} = (1, 0, 0, 0, 1, 1, 1)$$

Dopo Bob decifra y_1 ottenendo $x_1 = (1,0,0,0,1,1,0)$ (notiamo che $e_1 \neq e$ dovuto alla moltiplicazione per P^{-1}). Succesivamente Bob forma $x_0 = (1,0,0,0)$ (i primi quattro componenti di x_1).
Finalmente Bob calcola:

$$x = x_0 S^{-1} = (1,0,0,0)\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} = (1,1,0,1)$$

Che è il testo cifrato mandato da Alice.

References

[1] M. E. Hellman *"La crittografia a chiave pubblica"* da Le Scienze N. 136, dicembre 1979.

[2] W. Diffie e M. E. Hellman *"New directions in cryptography" IEEE Trans. Informat. Theory.* Nov. 1976.

[3] R. A. Rivest, A. Shamir, e I. Adleman *"A method for obtaining digital signatures and public-key cryptosystems"*, 1978.

[4] Bruce Schneier *"Applied Cryptography"* John Wiley & Sons, inc. 1993.

[5] James Nechvaltal *"Public key cryptography"* da Contemporary Cryptology IEEE PRESS 1992.