RSA

Anna Rita Strazioso

Fabrizio Piacenza

Valentina Ventriglia

ALGORITMO RSA

1) Introduzione

Passiamo a vedere in modo particolareggiato uno degli algoritmi asimmetrici più conosciuti ed utilizzati: **l'algoritmo RSA**.

Il nome dell'algoritmo deriva dalla prima lettera dei cognomi di coloro che lo inventarono nell'Aprile del 1977: Ronald I. **R**ivest, Adi **S**hamir e Leonard M. **A**dleman.

I campi di impiego dell'algoritmo sono diversi e variano passando dall'uso all'interno di programmi fino ad arrivare a costituire l'elemento portante di vere e proprie tecnologie di rete.

RSA è un cifrario a chiave pubblica che permette di cifrare un messaggio attraverso un procedimento che sfrutta le proprietà dei numeri primi.

2) Cifrari a chiave pubblica

2.1 Cosa sono?

I cifrari a chiave pubblica si basano su un concetto totalmente diverso da quello dei sistemi di cifratura con chiave simmetrica. La differenza principale consiste nel dover utilizzare una chiave per la cifratura (chiave detta pubblica), ma una chiave diversa per riportare il testo in chiaro (la chiave privata). La prima chiave può anche essere resa pubblica, dato che offre solo la possibilità di cifrare, ma non decifrare il messaggio. La differenza interna basilare consiste nell'utilizzo di una funzione unidirezionale, ovvero, la cui funzione inversa sia difficile da calcolare.

2.2 Perchè questa necessità?

Lo scopo per cui è stato inventato questo metodo, è da ricercarsi nella difficoltà di comunicare la chiave di decrittazione al destinatario. Nei metodi a chiave simmetrica, infatti, per comunicare la chiave, è necessario accordarsi di persona con il destinatario incontrandosi in un luogo sicuro, oppure utilizzare un canale di comunicazione sicuro. Ovvio però che se esistesse un canale di comunicazione sicuro non servirebbe neanche cifrare i messaggi! Ecco intervenire quindi i metodi a chiave pubblica che risolvono il problema.

2.3 Come si usa?

Immaginiamo che due persone (A e B), vogliano comunicare in segreto.
Entrambe sceglieranno una loro chiave segreta, ognuno per conto proprio: A sceglie "PassA" e B "PassB". Queste chiavi non le comunicheranno a nessuno. Tramite una funzione velocemente calcolabile, ma quasi impossibile da invertire, A e B generano la loro propria chiave pubblica che

potranno fornire a chiunque. Se adesso A vuole mandare un messaggio a B, è sufficiente utilizzare la chiave pubblica di B per cifrare il messaggio e spedirglielo. Solo B tramite la sua chiave segreta, potrà riconvertire in chiaro e velocemente, il messaggio di A. E' da notare che dopo la cifratura, nemmeno A (ovvero colui che cifra) ha più modo di riconvertire il messaggio in chiaro. Viceversa, se B vuole rispondere, è sufficiente che utilizzi la chiave pubblica di A per cifrare il messaggio.

2.4 Cosa si intende per funzione quasi impossibile da invertire?

Facciamo un esempio comprensibile a tutti: un numero elevato ad una potenza richiede un certo lavoro di calcolo, ma tornare indietro tramite la radice ennesima, necessita di un lavoro maggiore. Ancora meglio, calcolare il prodotto di due numeri primi impegna meno tempo che dover determinare i numeri primi conoscendo solo il loro prodotto. Nell'RSA, il più noto degli algoritmi a chiave pubblica, è proprio il prodotto di due numeri primi elevati ad essere utilizzato come funzione.

Tanto per chiarire con delle cifre, eseguire il prodotto di due numeri primi di cento cifre ciascuno, richiede con gli attuali calcolatori, un tempo dell'ordine di un milionesimo di secondo, mentre per fattorizzare un numero di duecento cifre decimali necessita qualche miliardo di anni.

Altre funzioni utilizzabili sono quelle di

1. *Diffie-Hellman*, considerato sicuro soprattutto nello scambio di chiavi simmetriche, quando usato con chiavi abbastanza lunghe preferibilmente di almeno 1024 bit. Sfrutta i logaritmi.
2. *Curve ellittiche*, è un algoritmo emergente, ma molto lento. È considerato estremamente sicuro ma non è stato ancora sottoposto agli stessi test a cui è stato sottoposto RSA.
3. *DSS (Digital Signature Standard)* usato principalmente per la firma digitale ed adottato dal governo USA.

2.5 Perchè non utilizzare esclusivamente questa categoria di cifratori?

Il sistema ha come tutte le cose, pregi e difetti.

- Il pregio è ovvio: il metodo è il più adatto che possa esistere per effettuare comunicazioni verso destinatari persino ignoti, dato che non necessita più concordare una password.
- Il difetto consiste nella sua lentezza di elaborazione dovuta ai pesanti calcoli matematici, tanto che viene generalmente utilizzato solo in sistemi composti, per trasmettere la password segreta di un altro sistema di crittografazione a chiave simmetrica (come ad esempio il DES). Non ha alcuno scopo, quindi, utilizzare un tale algoritmo per proteggere i dati sul proprio computer. Non solo: oltre alla lentezza, è

da tenere presente che a parità di lunghezza della chiave, gli algoritmi a chiave simmetrica sono sempre molto più sicuri.
- In questi sistemi la chiave non viene digitata, ma sta su hard disk e potrebbe essere rubata

2.6 Altre caratteristiche

Il sistema permette anche l'autentica del mittente e l'integrità del messaggio. Utilizzando un algoritmo di hashing, è possibile creare un'impronta del messaggio (detta hash o message-digest) e cifrarlo con la propria chiave privata. Il piccolo testo cifrato che ne deriva rappresenta la cosiddetta firma digitale. Dato che chiave privata e pubblica sono complementari, il destinatario, ricevuto il messaggio e conoscendo la chiave pubblica del mittente, sarà in grado di:

1. Riportare in chiaro il testo del messaggio
2. Riportare in chiaro il message-digest

È quindi chiaro che il messaggio proviene dal mittente la cui chiave pubblica è appena stata usata con successo per decifrare il message-digest (non fare confusione con il messaggio, che il destinatario decritta invece con la sua chiave privata). Per essere poi sicuri che il messaggio sia arrivato integro e senza manomissioni, basta ricreare un nuovo message-digest e controllare se è identico a quello allegato dal mittente.

Ma, tornando un passo indietro, se possiamo essere sicuri che un messaggio arriva realmente dal possessore di una data chiave privata utilizzata per la creazione del message-digest, come si possono conoscere realmente le generalità del mittente? Grazie ad appositi enti detti *Autorità di certificazione*, è possibile compilare un modulo con i propri dati e la propria chiave pubblica. L'ente verifica tali dati e se tutto corrispondente, rilascia un certificato firmato (sempre in digitale ovviamente) che funge da carta d'identità e può essere allegato ai propri messaggi.

2.7 Note sul message-digest

Come abbiamo già detto, per la creazione di un message-digest, vengono utilizzati algoritmi detti di *hash.* Questi algoritmi eseguono una serie di operazioni matematiche e logiche talvolta molto complesse il cui scopo è creare una specie di riassunto del messaggio (tipo checksum dei files) ottenendone una stringa più breve. Per la precisione, sono algoritmi one-way che producono, a partire da una stringa a lunghezza variabile, una stringa a lunghezza fissa (generalmente tra 64 e 255 bit) che è caratteristica della stringa data. Essendo l'algoritmo un one-way, si perdono le informazioni necessarie per invertire l'operazione, adempiendo l'altro requisito del digest che è appunto la sua irreversibilità in testo originale. La segretezza dell'algoritmo è fondamentale, dato che non dovrebbe neanche essere

possibile riuscire a creare un messaggio fasullo che abbia lo stesso digest, cosa semplicissima conoscendone il funzionamento. Ad oggi si sono messi a punto algoritmi di hashing che riducono la possibilità di avere 2 digests uguali da messaggi diversi in 1 su circa 4 miliardi di possibilità.

Gli algoritmi di hash più diffusi sono:

1. **MD5** (Message Digest Algorithm 5), sviluppato da RSA Data Security inc. Produce una stringa di hash a 128 bit da stringhe di lunghezza arbitraria. Largamente usato e ragionevolmente sicuro.
2. **SHA** (Secure Hash Algorithm), sviluppato dal NIST (National Institute of Standards and Technology) e dalla NSA (National Security Agency). Produce stringhe di hash a 160 bit da stringhe di lunghezza arbitraria. E` considerato abbastanza sicuro. Usato dal governo americano lo si trova solitamente associato all'utilizzo del metodo a chiave pubblica DSS.

2.8 Curiosità e considerazioni

Samuel Wagstaff, docente di informatica all'Università dell'Indiana, è riuscito a fattorizzare un numero di **167 cifre in centomila ore** di tempo computer. Il numero della prova era:

163790195580536623921741301 5
467044958392396568483270402 4
983781709239694686351321204 1
565096492260805419718247075 5
579714456896907387777297303 8
883717449030628887379284041

Questa notizia dovrebbe far riflettere: considerando che ad oggi si scoprono ancora nuovi algoritmi matematici per decrittare sempre più velocemente e che la potenza dei calcolatori aumenta vertiginosamente di mese in mese (e non parliamo dei computers dei laboratori segreti!), sarà una buona scelta affidare dati importantissimi ad un metodo che si basa esclusivamente sulla lentezza dei calcolatori attuali?

Vogliamo anche far notare che una chiave da 1024 bit in un sistema a chiave pubblica, vale circa quanto una a 64 bit di un sistema a chiave simmetrica a causa del fatto che nel sistema a chiave pubblica esiste sempre un legame tra chiave privata e segreta che permette di ridurre le combinazioni necessarie per trovare il codice di accesso.

Stabilita tale corrispondenza di sicurezza tra le lunghezze delle chiavi dei due sistemi, è interessante notare quando detto alla conferenza Crypto '93 (notare che sono già passati diversi anni), da M. Wiener del Bell Northern

Research, il quale ha descritto come con 1 milione di dollari sia realizzabile un chip speciale da 50 milioni di test al secondo che, in parallelo ad altri 57.000, può condurre un attacco con successo mediamente in 3,5 ore. Con un costo di 10 milioni di dollari il tempo si abbassa a 21 minuti, e con 100 milioni a disposizione, il codice è infranto in pochi secondi. Meditiamo!

Fatto sta che il commercio elettronico ha già iniziato a farne uso e qualche anno fa, il 5 Agosto 1997, il Consiglio dei Ministri Italiano ha approvato il regolamento di attuazione dell'art 15 della legge 57/97, nota anche come legge Bassanini-1, con il quale si stabilisce che l'originale di un documento può essere anche quello depositato su di un *file.* Tale documento su file ha valore probante sia sul contenuto sia sulla provenienza se corredato da firma elettronica legalmente riconosciuta.

2.10 Il funzionamento della crittografia a chiave pubblica

Diffie ed **Hellman** pensarono ad un sistema asimmetrico, basato su l'uso di due chiavi generate in modo che sia impossibile ricavarne una dall'altra. Le due chiavi vengono chiamate pubblica e privata: la prima serve per cifrare e la seconda per decifrare. Una persona che deve comunicare con un'altra persona non deve far altro che cifrare il messaggio con la chiave pubblica del destinatario, che una volta ricevuto il messaggio non dovrà fare altro che decifrarlo con la chiave segreta personale. Ogni persona con questo sistema possiede quindi una **coppia di chiavi**, quella pubblica può essere tranquillamente distribuita e resa di pubblico dominio perché consente solo di cifrare il messaggio, mentre quella privata deve essere conosciuta solo da una persona. In questo modo lo scambio di chiavi è assolutamente sicuro. Fino a questo punto sembrava andare tutto bene, ma bisognava trovare il modo di implementare matematicamente questo sistema, riuscire cioè a creare due chiavi per cui non fosse possibile dedurre quella privata conoscendo quella pubblica.
Il codice RSA si basa su un procedimento che utilizza i numeri primi e funzioni matematiche che è quasi impossibile invertire. Dati due numeri primi, e' molto facile stabilire il loro prodotto, mentre è molto più difficile determinare, a partire da un determinato numero, quali numeri primi hanno prodotto quel risultato dopo essere stati moltiplicati tra loro. In questo modo si garantisce quel principio di sicurezza alla base della crittografia a chiave pubblica infatti l'operazione di derivare la chiave segreta da quella pubblica è troppo complessa per venire eseguita in pratica

2.11 La forza dell'RSA

Il **codice RSA** viene considerato sicuro perché non è ancora stato trovato il modo per fattorizzare numeri primi molto grandi, che nel nostro caso significa riuscire a trovare **p** e **q** conoscendo **n.** . Nel corso degli anni l'algoritmo RSA ha più volte dimostrato la sua robustezza: in un esperimento del 1994,

coordinato da Arjen Lenstra dei laboratori Bellcore, per "rompere" una chiave RSA di 129 cifre, svelando il meccanismo con cui quella chiave generava messaggi crittografati, sono stati necessari 8 mesi di lavoro coordinato effettuato da 600 gruppi di ricerca sparsi in 25 paesi, che hanno messo a disposizione 1600 macchine da calcolo, facendole lavorare in parallelo collegate tra loro attraverso Internet.
Data la mole delle risorse necessarie per rompere la barriera di sicurezza dell'algoritmo RSA, è chiaro come un **attacco alla privacy** di un sistema a doppia chiave non sia praticamente realizzabile. Inoltre, nell'esperimento era stata utilizzata una chiave di 129 cifre mentre i programmi di crittografia attualmente a disposizione prevedono chiavi private con una "robustezza" che raggiunge e supera i 2048 bit, risultando quindi praticamente inattaccabili, visto anche che l'ordine di grandezza dei tempi necessari alla rottura di chiavi di questo tipo è esponenziale e passa in fretta da qualche giorno a qualche centinai di anni.

TEST DI PRIMALITA'

Un numero si dice **primo** quando è divisibile solo per se stesso e per 1; se possiede altri divisori, si dice che è **composto.** Escluso dunque il numero 2, tutti i numeri primi sono **dispari!!!**
Il **controllo della primalità** di un numero è la classificazione di quest'ultimo come primo o come composto.
La verifica di primalità di un numero n può essere effettuata in vari modi. La scelta è strettamente dipendente
dalla dimensione del numero:

- **Numeri piccoli ® Teorema di Wilson**
- **Numeri grandi ® Teorema di Fermat**

TEOREMA DI WILSON

Un numero **n** è primo se divide *senza resto*

- **(n-1)!+1**

Per **numeri piccoli**, il fattoriale è calcolabile:

- **n=5 ® 4!+1=25 ® 25/5=5 (resto=0)**
- **n=8 ® 7!+1=5041 ® 5041/8=630,125 (resto≠0)**
- **n=19 ® 18!+1=6402373705728001 ® 6402373705728001/19=336967037143579 (resto=0)**
- **n=51739721 ® 51739720! +1 = ® ??? troppo grande …**

Dunque per numeri troppo grandi il teorema di Wilson è IMPRATICABILE!!! … usiamo un altro algoritmo…**Teorema di Fermat**.

TEOREMA DI FERMAT

Un "veloce" algoritmo di controllo della ***quasi certa*** primalità di numeri con un elevato numero di cifre, si
basa sul cosiddetto ***piccolo teorema di Fermat***.
Il teorema stabilisce che:

- ***se p è un numero primo, allora per ogni numero naturale b appartenente all'intervallo aperto (0,p) è:***

$$b^p \pmod{p} = b$$

Un piccolo esempio:

- **I numeri *2*, *3* e *5* sono primi, e per essi è infatti:**

$$2 \triangleright 1^2 \pmod 2 = 1$$

$$3 \triangleright 1^3 \pmod 3 = 1; \quad 2^3 \pmod 3 = 2$$

$$5 \triangleright 1^5 \pmod 5 = 1; \quad 2^5 \pmod 5 = 2;$$

$$3^5 \pmod 5 = 3; \quad 4^5 \pmod 5 = 4$$

L' equivalenza logica del teorema è che:

- ***se esiste un numero naturale b appartenente all'intervallo aperto (0,p), per il quale:***

$$b^p \pmod p$$

  è diverso da b, allora p è un numero composto.

Un altro esempio per verificare quanto è stato appena detto:

- **Così ad esempio per il numero *p=4* è:**

$$4 \triangleright 1^4 \pmod 4 = 1; \; 2^4 \pmod 4 = 0; \; 3^4 \pmod 4 = 1$$

  per cui *4* è, come ci attendeva, composto.

E' lecito a questo punto chiedersi:

- ***se per ogni b, appartenente all'intervallo aperto (0,p), è:***

$$b^p \pmod p = b$$

  si può affermare che p è un numero primo ? Purtroppo no!

Esempio:

- **Esistono infatti numeri composti come**:
  - ***561* (il prodotto di *3*, *11* e *17*)**

- ***1729*** **(il prodotto di *7*, *13* e *19*)**

che, in relazione al piccolo teorema di Fermat, *si comportano come se fossero primi*. Siffatti numeri sono detti *numeri di Carmichael*.

Esistono anche numeri composti che, in relazione al piccolo teorema di Fermat, si comportano come se fossero primi, ***ma non per tutti i valori di b dell'intervallo aperto (0,p).***

Esempio:

- **Il numero composto *91* (il prodotto di *7* e *13*), è uno *pseudoprimo* in base 1 - 3 - 4 - 9 - 10 - 12 - 13 - 14 - 16 - 17 - 22 - 23 - 25 - 26 - 27 - 29 - 30 - 35 - 36 - 38 - 39 - 40 - 42 - 43 - 48 - 49 - 51 - 52 - 53 - 55 - 56 - 61 - 62 - 64 - 65 - 66 - 68 - 69 - 74 - 75 - 77 - 78 - 79 - 81 - 82 - 87 - 88 e 90. In tutto dunque *48 basi su 90*.**
- **Il numero composto *15* (il prodotto di *3* e *5*), è uno *pseudoprimo* in base 1 - 4 - 5 - 6 - 9 - 10 - 11 e 14. In tutto dunque *8 basi su 14*.**

Fortunatamente, se confrontati con i numeri primi, i numeri di Carmichael, ***numeri pseudoprimi in ogni base***, sono molto rari, non altrettanto i ***numeri pseudoprimi per alcune basi*** (come i numeri di cui sopra: *341*, *91* e *15*).

Esempio:

- **Scegliamo adesso a caso una base, ad esempio del numero pseudoprimo *15*. La probabilità che quest'ultimo *non venga riconosciuto come numero composto* dal test di Fermat vale: *8/14=0,57*. Se risottoponiamo al test di Fermat, con una nuova base, di nuovo il numero pseudoprimo *15*, la probabilità che quest'ultimo non venga *ancora una volta* riconosciuto come numero composto vale ora: *7/13=0,54*.**

  Se si continua, le probabilità che si ottengono sono:

  - ***6/12=0,5***
  - ***5/11=0,45***
  - ***4/10=0,4***

- ***3/9=0,3***
- ***2/8=0,25***
- ***1/7=0,14***

Che, come si vede, ***tendono a decrescere***.

E' importante a questo punto chiedersi: se il ***numero pseudoprimo 15*** non viene riconosciuto come numero composto per 4 volte consecutive, che probabilità ha di superare ***per la quinta volta*** il test di Fermat ?

La risposta è: ***0,07***=*0,57*0.54*0.5*0,45*.

Esclusi i numeri di Carmichael, risulta che :

- ***i numeri pseudoprimi per alcune basi, sono tali, per un numero di basi b, che, al più, è circa la metà di quelli compresi nell'intervallo aperto (0,p).***

Se dunque il controllo della primalità è su un numero *p* molto grande, scegliendo a caso *100* basi differenti, la probabilità che un numero pseudoprimo non venga riconosciuto come numero composto, ***nemmeno al 101-esimo test***, è all'incirca minore o uguale di:

$$\frac{1}{2^{100}} = \frac{1}{2^{10^{10}}} = \frac{1}{1024^{10}} \quad \text{circa uguale a} \quad \frac{1}{10^{30}}$$

Un numero dunque che non venga, ***dopo 100 tentativi***, riconosciuto come numero composto, dal test di Fermat, è ***quasi certamente un numero primo***.

GENERAZIONE DELLE CHIAVI

RSA è un cifrario a chiave pubblica che permette di cifrare un messaggio attraverso un procedimento che sfrutta le proprietà dei numeri primi. Supponiamo di avere come corrispondenti Kyle e Cartman, di cui il primo è il mittente del messaggio e il secondo è il destinatario.

Adesso vediamo come il destinatario genera le chiavi pubbliche e private per permettere a Kyle di cifrare il messaggio che gli vuole inviare. Procediamo per punti: (*Esempio con numeri grandi*).

Cartman genera le sue chiavi pubbliche e private

Cartman genera due numeri primi distinti **p** e **q** (su di essi si farà il testi di primalità per verificare che siano tali, si consiglia il piccolo teorema di Fermat perché i numeri sono abbastanza grandi), di valore consigliato maggiore di 10^100. Scegliamo:

- **p = 1069**
- **q = 1973**

Cartman genera n che è la prima chiave pubblica

Cartman calcola il prodotto dei due numeri primi ottenendo il numero **n**:

- **n = p x q ® n = 1069 x 1973 = 2109137**

Cartman calcola b che è la funzione di Eulero di n

Cartman dovrà, in pratica, calcolare il numero di naturali minori di n e primi con n:

- **b = _(n) = (p-1) x (q-1) ®**
  b = _(2109137) = (1069-1) x (1973-1) = 2106096

Una volta effettuati questi calcoli **p** e **q** vengono distrutti.

Cartman calcola e che è la seconda chiave pubblica

Cartman sceglie un numero **e** < _(n) che sia primo con b, ovvero tale che mcd(**e**,b) = 1. Calcoliamo (il valore è stato preso come tale da un esempio da internet, senza effettuare realmente e i calcoli):

- **e ® mcd(10001,2106096) = 1 SI**

- **e = 10001**

Cartman genera d che è la chiave privata

Cartman utilizzerà la versione estesa dell'algoritmo di Euclide (il teorema di Eulero sarebbe proibitivo), e calcola il più piccolo numero **d** tale che:

- **e*d = 1 mod _(n)**
- **d = 40433**

Per essere più precisi **d** è l'inverso di e nell'aritmetica di ordine b. I calcoli specifici in proposito saranno ripresi oltre con numeri più accessibili, adesso vogliamo solo dare un'idea dell'algoritmo.

Concludendo …

Quindi, dai calcoli effettuati da Cartman, ovvero il nostro destinatario del messaggio che genera le chiavi, risulterà:

- **Chiave pubblica (n,e) = (2109137,10001)**
- **Chiave privata (d) = (40433)**
- **Vorrei comunque sottolineare che p,q e b resteranno comunque sconosciuti a chiunque, infatti i primi due saranno distrutti dopo il calcolo della funzione di Eulero di n, ovvero dopo il calcolo di b.**

ESEMPIO

Ci concentreremo adesso su un esempio con numeri più piccoli per poter effettuare in dettaglio tutti i calcoli utilizzati per a generazione delle chiavi pubbliche e private.

Cartman genera le sue chiavi pubbliche e private

Cartman genera due numeri primi distinti **p** e **q** (su cui dovrà essere effettuato prima il test di primalità, si consiglia quello di Wilson perché i numero sono piccoli) e li moltiplica tra di loro ottenendo il numero **n** che viene reso pubblico, mentre **p** e **q** devono restare segreti.(**n = prima chiave pubblica**)

Sostituiamo un po' di numeri e facciamo i primi calcoli:

- **p = 5**
- **q = 11**
- **p x q = 5 x11 = 55 ➔ n = 55 (prima chiave pubblica)**

Cartman adesso calcola **b** che è la funzione di Eulero di **n**.

Proseguiamo con i calcoli:

- **b = _(n) = (p-1) x (q-1) (Il numero b deve restare segreto)**
- **_(55) = (5 - 1) x (11 - 1) = 4 x 10 = 40 ➔ b = 40**

Cartman calcola il primo intero **e** che sia primo con **b** (cioè che non abbia divisori in comune, ovvero MCD(e, b) = 1). (**e = seconda chiave pubblica**)

Per effettuare questo calcolo si è proceduto per tentativi

- **e = 2 ➔ MCD(2, 40) = 2 NO**
- **e = 3 ➔ MCD(3, 40) = 1 SI ➔ e = 3 (seconda chiave pubblica)**

Adesso procediamo con la parte un po' più difficile, ovvero il calcolo della chiave privata **d**.
Cartman calcola il numero **d** inverso di **e** nell'aritmetica finita di ordine **b**, che è il più piccolo valore x per cui sia **e*d** mod **b** = 1; il numero **d** é la chiave per decifrare e *deve restare segreto*. Dato che i numeri in questione sono piccoli, abbiamo usato per risolvere l'inverso, il *teorema di Eulero*. Vedremo però nei calcoli che si è fatto uso anche della *funzione di Eulero* per calcolare **_(b)**.
(**d = chiave privata**)

Vediamo come:

- **Si calcola il più piccolo numero d, inverso di e nell'aritmetica di ordine b:**
- **Devo trovare in Z_{40} $[3]^{-1}$, per farlo utilizzo questa formula:**
  - **Z_b, $[e]^{-1} = [\ e^{-(b)\ -\ 1}]$**
- **Calcolo la funzione di Eulero per _(b):**
- **_(b) = _(5)* _(4) ➔ _(40) = 4 * 2 = 8 ➔ _(40) = 8**
- **Utilizzando il teorema di Eulero posso calcolare d:**
- **Z_{40} , $[3]^{-1} = [\ 3^{-(40)\ -\ 1}]$ ➔ $[\ 3^{8\ -\ 1}] = 2187$ ➔ $[27]_{40}$**

Concludendo, adesso siamo in possesso sia della chiave pubblica che di quella privata, che sono:

- **Chiave pubblica (n,e) = (55,3)**
- **Chiave privata (d) = (27)**

OSSERVAZIONI E SPECCHIETTO RIASSUNTIVO

Il codice RSA si basa su un procedimento che utilizza i numeri primi e funzioni matematiche che è quasi impossibile invertire. Dati due numeri primi, e' molto facile stabilire il loro prodotto, mentre è molto più difficile determinare, a partire da un determinato numero, quali numeri primi hanno prodotto quel risultato dopo essere stati moltiplicati tra loro. In questo modo si garantisce quel principio di sicurezza alla base della crittografia a chiave pubblica. Infatti, l'operazione di derivare la chiave segreta da quella pubblica è troppo complessa per venire eseguita in pratica.

In linea di principio, l'analisi dell'algoritmo potrebbe essere facilmente suddivisa in due parti: la *generazione della coppia di chiavi* e *l'utilizzo delle stesse*.

La prima parte, ***la generazione della coppia di chiavi***, viene solitamente effettuata in questo modo:

- **Vengono scelti due numeri primi p, q molto grandi (sono consigliati valori maggiori di 10^100);**
- **Viene effettuato su di essi il test di primalità (se sono piccoli utilizzeremo il teorema di Wilson, altrimenti il piccolo teorema di Ferma)**
- **Viene calcolato n = p x q, e la funzione di Eulero _(n) = (p - 1) x (q - 1) dopo di che i due primi p, q vengono eliminati;**
- **Si sceglie un intero e minore di _(n) e primo con esso, tale che mcd(e, _(n))=1;**
- **Utilizzando la versione estesa dell'algoritmo di Euclide (o utilizzando il teorema di Eulero se il numero è sufficientemente piccolo) viene calcolato l'intero d così da avere e * d = 1 mod _(n), o ancora meglio, in modo da ottenere l'inverso di e nell'aritmetica di ordine _(n);**
- **Vengono resi pubblici i valori e, n che costituiscono la chiave pubblica e mantenuto segreto d.**

PASSI PER CIFRARE E DECIFRARE IL MESSAGGIO: VADO!

La regola di **cifratura**, come quella di **decifratura**, usa dei numeri decimali. Poiché il messaggio da cifrare è una sequenza di lettere dobbiamo ottenere,

attraverso una traduzione, una sequenza di numeri da sottoporre successivamente alla regola di cifratura.

TRADUZIONE

Il messaggio va **tradotto in una sequenza di numeri**. Inizialmente bisogna accordarsi sulla modalità di traduzione: potrebbe essere per esempio il codice ASCII dei singoli caratteri, ma così il cifrario degenererebbe in un banale cifrario monoalfabetico.
Nel nostro esempio associamo ad ogni lettera dell'alfabeto un numero ottenendo così una delle tante tavole degli alfabeti. Per semplicità associamo:

A=01	H=08	O=15	V=22
B=02	I=09	P=16	W=23
C=03	J=10	Q=17	X=24
D=04	K=11	R=18	Y=25
E=05	L=12	S=19	Z=26
F=06	M=13	T=20	_=27
G=07	N=14	U=21	!=28

A questo punto confrontiamo ogni lettera da inviare con la tabella, al fine di ottenere il numero corrispondente:

- **V=22, A=01, D=04, O=15, !=28**

Il numero da cifrare sarà pertanto quello costituito da tutti i numeri ottenuti a seguito della traduzione:

- **2201041528**

Affiche il codice decifrato dal destinatario sia uguale a quello cifrato dal mittente occorre che il **numero da cifrare abbia un valore minore rispetto a quello della chiave pubblica n** (n=55). Il numero ottenuto a seguito della traduzione è:

- **2201041428>55**

pertanto non può essere cifrato senza un opportuna modifica. Possiamo operare la scomposizione in blocchi del numero ottenuto.

DIVISIONE IN BLOCCHI

Dobbiamo a questo punto **formare dei blocchi, con k elementi, tali che il valore di ogni blocco sia minore o uguale al valore della chiave pubblica n**.
La chiave pubblica n usata in questo esempio è 55. Poiché il nostro alfabeto non arriva fino al valore 55(assume come valore massimo 28), possiamo fare blocchi di due numeri.

- **2201041528 ➔ 22 01 04 15 28**

Se non avessimo ricordato il valore numerico massimo trovato nella tabella degli alfabeti avremmo potuto fare dei tentativi. Supponendo di poter costruire blocchi con 3 numeri(k=3) saremmo dovuti andare a confrontare ogni blocco ottenuto, con il valore della chiave n. Già analizzando il primo blocco 220 però, avremmo scoperto che quest'ultimo non soddisfa la relazione di minoranza rispetto alla chiave pupplica n (220>55). A questo punto avremmo potuto costruire blocchi con k=2 e avremmo scoperto che il valore di ogni blocco ottenuto dalla scomposizione risulta essere minore al 55. Pertanto formare blocchi con due valori risulta essere la scelta più opportuna. Nel caso in cui l'ultimo blocco non fosse stato costituito da k numeri, avremmo inserito una sequenza di 27(poiché nella tabella degli alfabeti utilizzata nella traduzione, il numero 27 corrisponde a " ").
A questo punto **siamo pronti per la cifratura**

CIFRATURA

Per la cifratura si utilizza la formula:

- **c=m^e mod n**

dove **e** ed **n** sono le chiavi pubbliche del destinatario (**n=55,e=3**) mentre **m** è il valore del blocco da cifrare.
A questo punto applichiamo tale formula a tutti i blocchi ottenuti:

- **c1=22^3 mod 55=33**
- **c2=1^3 mod 55=1**
- **c3=4^3 mod 55=9**
- **c4=15^3 mod 55=20**
- **c5=28^3 mod 55=7**

A questo punto abbiamo ottenuto il messaggio cifrato:

- **33 01 09 20 07**

e **può essere trasmesso**. Nel momento in cui il destinatario lo riceve, la prima cosa che deve fare è **decifrarlo**.

DECIFRATURA

Il messaggio da decifrare è:

- **33 01 09 20 07**

Solo il legittimo destinatario può decifrare il messaggio perchè solo lui è a conoscenza della sua chiave segreta. Infatti quest'ultima viene utilizzata all'interno della regola di decifratura.

Per la decifratura si usa la formula:

- **m=c^d mod n**

dove **d** è la chiave segreta del destinatario, **n** è la chiave pubblica e **c** è il valore del blocco da decifrare(**n=55, d=27**).

- **m1=33^27 mod 55=22**
- **m2=1^27 mod 55=1**
- **m3=9^27 mod 55=4**
- **m4=20^27 mod 55=15**
- **m5=7^27 mod 55=28**

Il messaggio decifrato:

- **22 01 04 15 28**

A questo punto il destinatario non deve fare altro che effettuare la traduzione utilizzando la tabella degli alfabeti utilizzata per la traduzione iniziale.

TRADUZIONE

Confrontiamo i valori ottenuti con la tabella di traduzione utilizzata

- **22=V, 1=A, 4=D, 15=O, 28=! ➔VADO!**

Nel caso in cui inizialmente avessimo effettuato **la traduzione considerando il codice ASCII** avremmo ottenuto:

- **V=01010110**
- **A=01000001**
- **D=01000100**
- **O=01001111**
- **!=00100000**

sequenza da cifrare:

- **0101011001000001010001000100111100100000**

Questa sequenza assume, tradotta in decimale, un valore sicuramente maggiore di **n** (n=55).
Anche in questo caso dobbiamo optare per una scomposizione della sequenza in blocchi.
Come fare?
55 in codice binario è rappresentato da 6 bit, pertanto affinche il valore di ogni blocco sia minore di 55, basta scomporre la sequenza in blocchi da 5 bit.

- **01010 11001 00000 10100 01000 10011 11001 00000**

Nel caso in cui **l'ultimo blocco non fosse stato completo** si sarebbero dovuti **aggiungere degli zeri** al fine di completare anche l'ultimo blocco. In questo modo tutti i blocchi saranno costituiti da **k** bit (nel nostro esempio k=5).

Un ulteriore considerazione da fare è quella di considerare che le formule di cifratura e di decifratura operano con numeri decimali e non con numeri binari pertanto occorre tradurre ogni numero binario nel corrispettivo numero decimale.

- **01010 =10**
- **11001=25**
- **00000=0**
- **10100=20**
- **01000=8**
- **10011=19**
- **11001=25**
- **00000=0**

La sequenza da decifrare sarà pertanto:

- **10 25 0 20 8 19 25 0**

Per la cifratura si esegue la stessa operazione effettuata per l'esempio precedente, così come per la decifratura.

CIFRATURA:

- **C(1)=10^3 mod 55=10**
- **C(2)=25^3 mod 55=5**
- **C(3)=0^3 mod 55=0**
- **C(4)=20^3 mod 55=25**
- **C(5)=8^3 mod 55=17**
- **C(6)=19^3 mod 55=39**
- **C(7)=25^3 mod 55=5**
- **C(8)=0^3 mod 55=0**

DECIFRATURA:

- **m(1)=10^27 mod 55=10**
- **m(2)=5^27 mod 55=25**
- **m(3)=0^27 mod 55=0**
- **m(4)=25^27 mod 55=20**
- **m(5)=17^27 mod 55=8**
- **m(6)=39^27 mod 55=19**
- **m(7)=5^27 mod 55=25**
- **m(8)=0^27 mod 55=0**

Per leggere il messaggio bisognerà tradurlo. La tabella del codice ASCII considera numeri binari pertanto la prima cosa da fare sarà quella di convertire in binario i numeri decimali ottenuti dalla decifratura:

- **10=01010**
- **25=11001**
- **0=00000**
- **20=10100**
- **8=01000**
- **19=10011**
- **25=11001**
- **0=00000**

La **sequenza risultante** sarà costituita da **tutti i blocchi ottenuti dalla conversione in binario**

- **0101011001000001010001000100111100100000**

A questo punto si dovranno formare blocchi da 8 numeri(nel sistema ASCII ogni numero è rappresentato da 8 bit), ottenendo così:

- **01010110 01000001 01000100 01001111 00100000**

Per completare la traduzione confrontiamo ogni blocco con la tabella del codice ASCII al fine di ottenere le lettere corrispondenti.

Confrontando tali blocchi con la tavola dei codici ASCII ritroveremo il messaggio **VADO!**

DIMOSTRAZIONE

DIMOSTRAZIONE:IL MESSAGGIO OTTENUTO E' VERAMENTE QUELLO INIZIALE?

Poiché per la cifratura:

- **c = m^e mod n**

e per la decifratura:

- **m* = c^e mod n**

allora dobbiamo dimostrare che:

- **m*=m^(e*d) mod n** [1]

Iniziamo ricordando che d è stato scelto tale che:

- **e*d = 1 mod _(n)**

ciò significa che esiste un intero positivo **k** tale che:

- **e*d=1+ k* _(n)** [2]

Visto che **n=p*q** con p e q primi, allora:

- **_(n)=(p-1)(q-1)** [3]

Sostituendo cosi la [3] nella [2] otteniamo:

- **e*d=1+k* _(n) =1+k*(p-1)(q-1)** [4]

Il valore ottenuto dalla [4] possiamo sostituirlo ora nella [1]:

- **m*=m^(1+k*(p-1)(q-1)) mod p*q**

Per il **teorema di Eulero** tale valore è proprio uguale ad **m** quindi **m*=m**.
Abbiamo così dimostrato il teorema e possiamo così affermare che il messaggio ricevuto dal destinatario è proprio quello mandato dal mittente.