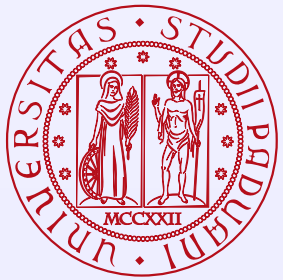
RSA: firma digitale e attacchi

Alessandro Languasco

Dipartimento di Matematica
Università di Padova

Padova, 28/11/2014



Crittografia

La crittografia studia i metodi che possono essere usati per inviare informazioni in forma celata, in modo tale che solamente il destinatario autorizzato possa rimuovere l'impedimento e leggere il messaggio in forma chiara

Schema generale di un crittosistema

$$\mathfrak{M} = \{\text{messaggi in chiaro}\}$$

$$\mathfrak{C} = \{\text{messaggi cifrati}\}$$

sono gli insiemi dei messaggi in forma chiara e celata.

Diremo *trasformazione crittografica* una funzione **iniettiva**

$$f : \mathfrak{M} \to \mathfrak{C}.$$

La funzione di decifratura è la funzione inversa f^{-1} definita :

$$\mathfrak{M} \xrightarrow{f} f(\mathfrak{M}) \xrightarrow{f^{-1}} \mathfrak{M}.$$

(l'iniettività di f evita ambiguità di decifrazione)

Schema generale di un crittosistema/2

Queste funzioni vengono anche chiamate:

$$f\text{: } \textit{funzione di cifratura}$$
$$f^{-1}\text{: } \textit{funzione di decifratura}$$

Esse, in generale sono dipendenti da alcuni parametri detti *chiavi*.

Crittosistema: quaterna $(\mathfrak{M}, \mathfrak{C}, f, f^{-1})$ in cui f è una trasformazione crittografica.

RSA in breve

(1) ogni utente sceglie in modo casuale due primi p, q distinti ed estremamente grandi (diciamo di circa 300 cifre decimali ciascuno) e pone $n = pq$;

RSA in breve

(1) ogni utente sceglie in modo casuale due primi p, q distinti ed estremamente grandi (diciamo di circa 300 cifre decimali ciascuno) e pone $n = pq$;

(2) calcola $\varphi(n) = (p-1)(q-1) = \operatorname{card} \mathbb{Z}_n^*$;

RSA in breve

(1) ogni utente sceglie in modo casuale due primi p, q distinti ed estremamente grandi (diciamo di circa 300 cifre decimali ciascuno) e pone $n = pq$;

(2) calcola $\varphi(n) = (p-1)(q-1) = \operatorname{card} \mathbb{Z}_n^*$;

(3) sceglie in modo casuale un intero e tale che $1 < e < \varphi(n)$ e $(e, \varphi(n)) = 1$ (e coprimo con $\varphi(n)$);

RSA in breve

(1) ogni utente sceglie in modo casuale due primi p, q distinti ed estremamente grandi (diciamo di circa 300 cifre decimali ciascuno) e pone $n = pq$;

(2) calcola $\varphi(n) = (p-1)(q-1) = \operatorname{card} \mathbb{Z}_n^*$;

(3) sceglie in modo casuale un intero e tale che $1 < e < \varphi(n)$ e $(e, \varphi(n)) = 1$ (e coprimo con $\varphi(n)$);

(4) calcola $d \equiv e^{-1} \mod \varphi(n)$ (per lui è "facile" perché conoscendo sia p che q può calcolare facilmente $\varphi(n)$ e usare l'Algoritmo Euclideo Esteso per determinare d conoscendo e e $\varphi(n)$).

RSA in breve

(1) ogni utente sceglie in modo casuale due primi p, q distinti ed estremamente grandi (diciamo di circa 300 cifre decimali ciascuno) e pone $n = pq$;

(2) calcola $\varphi(n) = (p-1)(q-1) = \operatorname{card} \mathbb{Z}_n^*$;

(3) sceglie in modo casuale un intero e tale che $1 < e < \varphi(n)$ e $(e, \varphi(n)) = 1$ (e coprimo con $\varphi(n)$);

(4) calcola $d \equiv e^{-1} \mod \varphi(n)$ (per lui è "facile" perché conoscendo sia p che q può calcolare facilmente $\varphi(n)$ e usare l'Algoritmo Euclideo Esteso per determinare d conoscendo e e $\varphi(n)$).

In quanto sopra è importante notare che:

(a) le scelte devono essere *casuali*;

(b) il calcolo di d è un sottoprodotto di $(e, \varphi(n)) = 1$.

RSA: parametri

Per *ogni utente* saranno quindi definite le seguenti quantità:

RSA: parametri

Per *ogni utente* saranno quindi definite le seguenti quantità:
Chiave di Cifratura (pubblica):

$$K_E = (n, e).$$

RSA: parametri

Per *ogni utente* saranno quindi definite le seguenti quantità:
Chiave di Cifratura (pubblica):

$$K_E = (n, e).$$

Chiave di Decifratura (privata):

$$K_D = d.$$

RSA: parametri

Per *ogni utente* saranno quindi definite le seguenti quantità:
Chiave di Cifratura (pubblica):

$$K_E = (n, e).$$

Chiave di Decifratura (privata):

$$K_D = d.$$

Funzione di cifratura: è la funzione $f : \mathbb{Z}_n \to \mathbb{Z}_n$ data da

$$f(M) \equiv M^e \bmod n.$$

RSA: parametri

Per *ogni utente* saranno quindi definite le seguenti quantità:
Chiave di Cifratura (pubblica):

$$K_E = (n, e).$$

Chiave di Decifratura (privata):

$$K_D = d.$$

Funzione di cifratura: è la funzione $f : \mathbb{Z}_n \to \mathbb{Z}_n$ data da

$$f(M) \equiv M^e \bmod n.$$

Funzione di decifratura: è la funzione $f^{-1} : \mathbb{Z}_n \to \mathbb{Z}_n$ data da

$$f^{-1}(C) \equiv C^d \bmod n.$$

Sicurezza del sistema RSA

I punti fondamentali che assicurano la sicurezza del sistema sono:

Sicurezza del sistema RSA

I punti fondamentali che assicurano la sicurezza del sistema sono:

(1) f è legata alla “facilità” computazionale di calcolare i primi p e q;

Sicurezza del sistema RSA

I punti fondamentali che assicurano la sicurezza del sistema sono:

(1) f è legata alla "facilità" computazionale di calcolare i primi p e q;

(2) *violare* il sistema (cioè calcolare $f^{-1}(c)$ senza conoscere K_D) è legato alla capacità di calcolare d conoscendo solo e e n. Per fare ciò bisogna essenzialmente essere capaci di calcolare $\varphi(n)$ conoscendo n (ma *senza* conoscere la sua fattorizzazione).

Sicurezza del sistema RSA

I punti fondamentali che assicurano la sicurezza del sistema sono:

(1) f è legata alla "facilità" computazionale di calcolare i primi p e q;

(2) *violare* il sistema (cioè calcolare $f^{-1}(c)$ senza conoscere K_D) è legato alla capacità di calcolare d conoscendo solo e e n. Per fare ciò bisogna essenzialmente essere capaci di calcolare $\varphi(n)$ conoscendo n (ma *senza* conoscere la sua fattorizzazione).

Ma

conoscere $\varphi(n)$ *è computazionalmente equivalente a conoscere la fattorizzazione di* n.

Sicurezza del sistema RSA

I punti fondamentali che assicurano la sicurezza del sistema sono:

(1) f è legata alla "facilità" computazionale di calcolare i primi p e q;

(2) *violare* il sistema (cioè calcolare $f^{-1}(c)$ senza conoscere K_D) è legato alla capacità di calcolare d conoscendo solo e e n. Per fare ciò bisogna essenzialmente essere capaci di calcolare $\varphi(n)$ conoscendo n (ma *senza* conoscere la sua fattorizzazione).

Ma

conoscere $\varphi(n)$ *è computazionalmente equivalente a conoscere la fattorizzazione di* n.

Violare il sistema RSA equivale essenzialmente a fattorizzare n con complessità equivalente a quella con cui si prova la primalità di p, q.

Cenni su algoritmi di primalità e fattorizzazione

Per *algoritmo di primalità* intenderemo *un insieme finito di calcoli che (senza cercare i fattori di n) prova il fatto che n è primo*.

Cenni su algoritmi di primalità e fattorizzazione

Per *algoritmo di primalità* intenderemo *un insieme finito di calcoli che (senza cercare i fattori di n) prova il fatto che n è primo*.
Migliore algoritmo di primalità oggi noto: *Agrawal, Kayal, Saxena* (2002); ha complessità computazionale (nella versione migliorata di Lenstra-Pomerance) pari a

$$O\left((\log n)^{6+\varepsilon}\right)$$

Cenni su algoritmi di primalità e fattorizzazione

Per *algoritmo di primalità* intenderemo *un insieme finito di calcoli che (senza cercare i fattori di n) prova il fatto che n è primo*.
Migliore algoritmo di primalità oggi noto: *Agrawal, Kayal, Saxena* (2002); ha complessità computazionale (nella versione migliorata di Lenstra-Pomerance) pari a

$$O\left((\log n)^{6+\varepsilon}\right)$$

Algoritmo di *fattorizzazione*: un algoritmo che *fornisce almeno un fattore primo del numero n*

Cenni su algoritmi di primalità e fattorizzazione

Per *algoritmo di primalità* intenderemo *un insieme finito di calcoli che (senza cercare i fattori di n) prova il fatto che n è primo*.
Migliore algoritmo di primalità oggi noto: *Agrawal, Kayal, Saxena* (2002); ha complessità computazionale (nella versione migliorata di Lenstra-Pomerance) pari a

$$O\left((\log n)^{6+\varepsilon}\right)$$

Algoritmo di *fattorizzazione*: un algoritmo che *fornisce almeno un fattore primo del numero n*
Migliore algoritmo di fattorizzazione: Pollard-Lenstra-Lenstra (1993); ha complessità che si *congettura* essere pari a

$$O\left(e^{c\sqrt[3]{\log n(\log\log n)^2}}\right), \text{ dove } c > 0 \text{ è una costante fissata,}$$

Cenni su algoritmi di primalità e fattorizzazione

Come vedete, per i metodi di fattorizzazione si sanno fornire stime del caso peggiore, cioè delle maggiorazioni, mentre, per provare l'unidirezionalità della trasformazione crittografica di RSA, servirebbero opportune minorazioni.

Pertanto, poiché determinare minorazioni significative della complessità del caso medio della fattorizzazione è un problema aperto, il sistema RSA può essere considerato a chiave pubblica soltanto da un punto di vista congetturale.

Attacchi a RSA: generalità

- I numeri primi sono sufficientemente numerosi da rendere RSA realizzabile; se i primi fossero molto rari, la scelta dei parametri sarebbe molto difficile, ed il problema di scomporre un intero nei suoi fattori primi molto facile.

Attacchi a RSA: generalità

- I numeri primi sono sufficientemente numerosi da rendere RSA realizzabile; se i primi fossero molto rari, la scelta dei parametri sarebbe molto difficile, ed il problema di scomporre un intero nei suoi fattori primi molto facile.

- La scelta dei primi p, q deve essere fatta accuratamente perché esistono algoritmi di fattorizzazione che hanno una buona complessità computazionale nei seguenti casi:

Attacchi a RSA: generalità

- I numeri primi sono sufficientemente numerosi da rendere RSA realizzabile; se i primi fossero molto rari, la scelta dei parametri sarebbe molto difficile, ed il problema di scomporre un intero nei suoi fattori primi molto facile.

- La scelta dei primi p, q deve essere fatta accuratamente perché esistono algoritmi di fattorizzazione che hanno una buona complessità computazionale nei seguenti casi:
  - numeri che hanno fattori primi piccoli ("Trial Division method");
  - numeri aventi i fattori tra loro vicini ("Fermat-Lehmer method");
  - numeri n tali che $n+1$ o $n-1$ siano formati da soli fattori primi piccoli ("Williams-Lucas method").

Ordini di grandezza

Il NIST attualmente consiglia come minimo livello di sicurezza in RSA di usare chiavi n di lunghezza 2048 bit (circa 617 cifre decimali) durante i processi di scambio chiavi. Chiaramente maggiore sicurezza può essere ottenuta utilizzando chiavi di 3072 bit (circa 925 cifre decimali) o 4096 bit (circa 1234 cifre decimali) per applicazioni più “sensibili” quali la generazione delle chiavi necessarie al funzionamento di un Ente certificatore.

Ordini di grandezza

Il NIST attualmente consiglia come minimo livello di sicurezza in RSA di usare chiavi n di lunghezza 2048 bit (circa 617 cifre decimali) durante i processi di scambio chiavi. Chiaramente maggiore sicurezza può essere ottenuta utilizzando chiavi di 3072 bit (circa 925 cifre decimali) o 4096 bit (circa 1234 cifre decimali) per applicazioni più "sensibili" quali la generazione delle chiavi necessarie al funzionamento di un Ente certificatore.
IMPORTANTE: La chiave di un utente individuale non dovrebbe avere durata superiore a due anni (passato questo lasso di tempo essa va nuovamente generata eventualmente tenendo conto di nuove raccomandazioni).

Ordini di grandezza

- un foglio di carta da 0,1 mm piegato in due 42 **volte** ha spessore maggiore della **distanza terra-luna**
  ($2^{42} = 4398046511104$; lo spessore è di 4398046511104 decimillimetri ossia 439804.6511104 km; distanza “media” Terra-Luna = 384.000 km (356.410 km al perigeo e 406.700 km all’apogeo))
- un personal computer trova primi p e q di circa 300 cifre decimali (circa 1024 bit) in pochi **secondi**; un supercomputer impiega **mesi** per fattorizzare $N = pq$ di circa 600 cifre decimali

FIRMA DIGITALE: schema generale

A e B utenti di un sistema a chiave pubblica.
f_A e f_B sono pubbliche; f_A^{-1} e f_B^{-1} sono segrete; assumiamo inoltre f_A e f_B funzioni surgettive (oltre che iniettive).
A invia un messaggio M a B : $f_B(M)$.
A, certifica la propria identità a B: $f_B(f_A^{-1}(s_A))$. ($f_A^{-1}(s_A)$: la firma digitale di A ; s_A un nome convenzionale di A (con numero progressivo, tempo di spedizione, numero IP macchina speditrice, ...).

- B decifra il messaggio: $f_B^{-1}(f_B(M)) = M$.
- B applica poi $f_A f_B^{-1}$; $f_A f_B^{-1}(f_B f_A^{-1}(s_A)) = s_A$.
- solo A può aver firmato in tale modo (solo lui conosce f_A^{-1}).

Firma digitale/2

Problemi:

1) Connessione tra persona e funzione di decifratura (non è un problema crittografico ma generale); contromisura: figura super-partes (*Ente Certificatore* delle chiavi pubbliche (B può controllare la firma di A)).
2) un intruso può identificare $f_A^{-1}(s_A)$ utilizzando un numero alto di messaggi intercettati (contromisura: firma dipendente dal messaggio).

Firma digitale con RSA

A lavora su $\mathbb{Z}_{n_A}$ e B lavora su $\mathbb{Z}_{n_B}$, ed in generale $n_A \neq n_B$.
Supponiamo dunque che l'utente A, con chiave pubblica (n_A, e_A) e funzione crittografica f_A voglia convincere della propria identità l'utente B, con chiave pubblica (n_B, e_B) e funzione crittografica f_B. Per raggiungere questo scopo, l'utente A sceglie una "firma digitale" s_A che rende pubblica: in pratica A sceglie $s_A \in \mathbb{Z}_{n_A}$. Per convincere B della propria identità invia una forma crittografata della firma:

$$m_A = f_B\left(f_A^{-1}(s_A)\right) \text{ se } n_A < n_B; \quad m_A = f_A^{-1}\left(f_B(s_A \bmod n_B)\right) \text{ se } n_A > n_B,$$

Per assicurarsi dell'identità di A, B calcola

$$f_A\left(f_B^{-1}(m_A)\right) \quad \text{se } n_A < n_B; \qquad f_B^{-1}\left(f_A(m_A)\right) \quad \text{se } n_A > n_B.$$

Tutto ciò funziona perché solo A può calcolare f_A^{-1}, e solo B può calcolare f_B^{-1}.

Firma dipendente dal messaggio

- firma digitale: $f_A^{-1}(h(M))$ (h funzione hash)
- $h(M)$ (detta *impronta* di M) è una sequenza di bit di lunghezza fissata (160 bit)
- h non permette di risalire a M conoscendo solamente $h(M)$.
- la probabilità che $h(M) = h(M')$ con $M \neq M'$ sia minore di 10^{-50}.

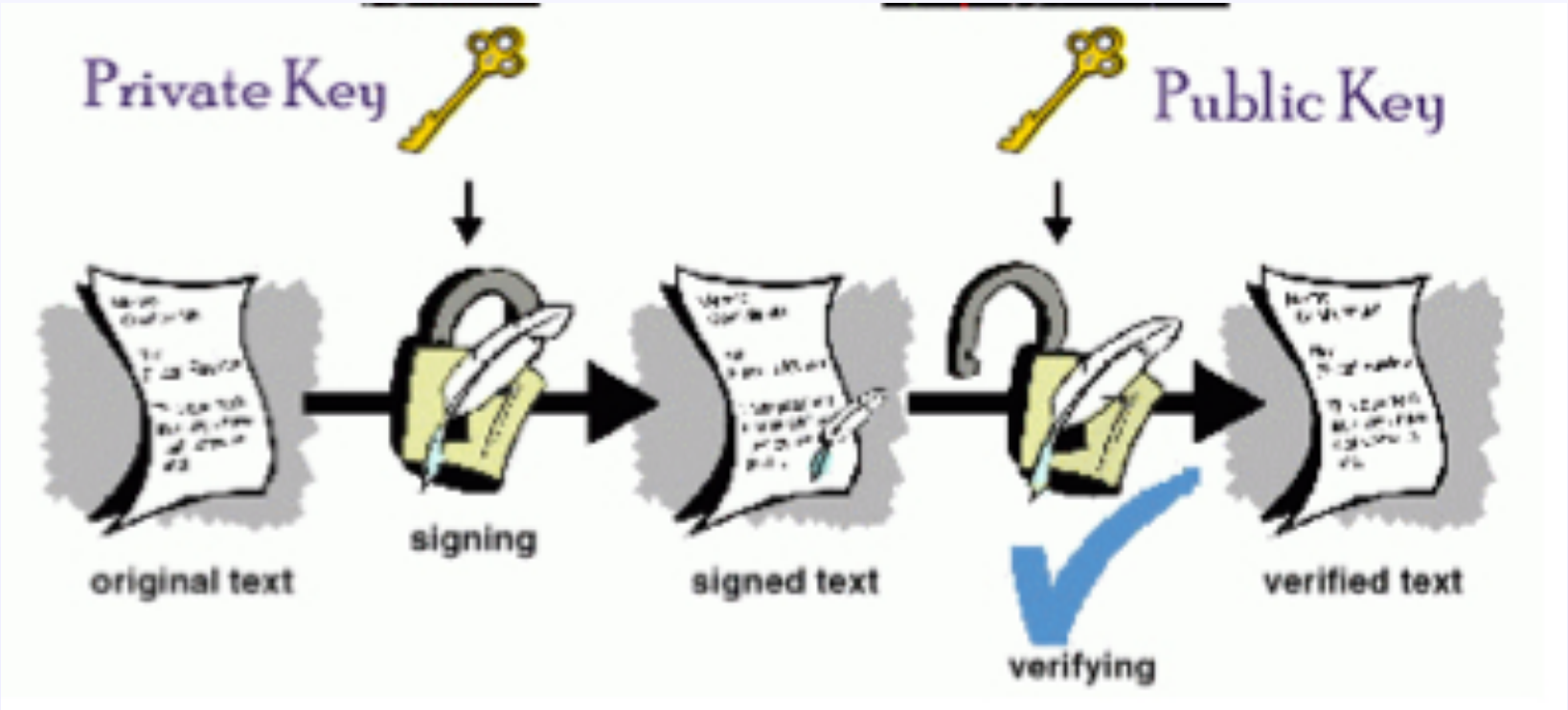
La funzione h è unica per tutti gli utenti del crittosistema (h pubblica). Verifica della firma: B controlla che il mittente sia A e ottiene $h(M)$; B ricalcola l'impronta di M (h pubblica) (conseguenza: probabile integrità del messaggio).

Marcatura Temporale

L'Ente Certificatore consente *marcatura temporale* (analoga all'autenticazione di un documento) del messaggio stesso.

- Per datare un messaggio, A invia all'Ente Certificatore la quantità $f_B(f_A^{-1}(h(M)))$.
- l'Ente Certificatore aggiunge al data e l'ora T, applica la propria chiave privata e reinvia il tutto ad A: $f_E^{-1}(f_B(f_A^{-1}(h(M))), T))$.
- A allega la firma digitale al messaggio M e spedisce a B la quantità: $f_B(M, f_E^{-1}(f_B(f_A^{-1}(h(M))), T))$.

Crittografia moderna: firma digitale



Come funziona una carta bancomat “sicura”

Il problema è quello di doversi identificare presso lo sportello bancomat. Ciò viene realizzato dimostrando di conoscere la propria *chiave segreta*, ma, chiaramente, ciò deve essere realizzato *SENZA* mandarla alla banca.

Come funziona una carta bancomat “sicura”/2

La crittografia a chiave pubblica consente di risolvere il problema:

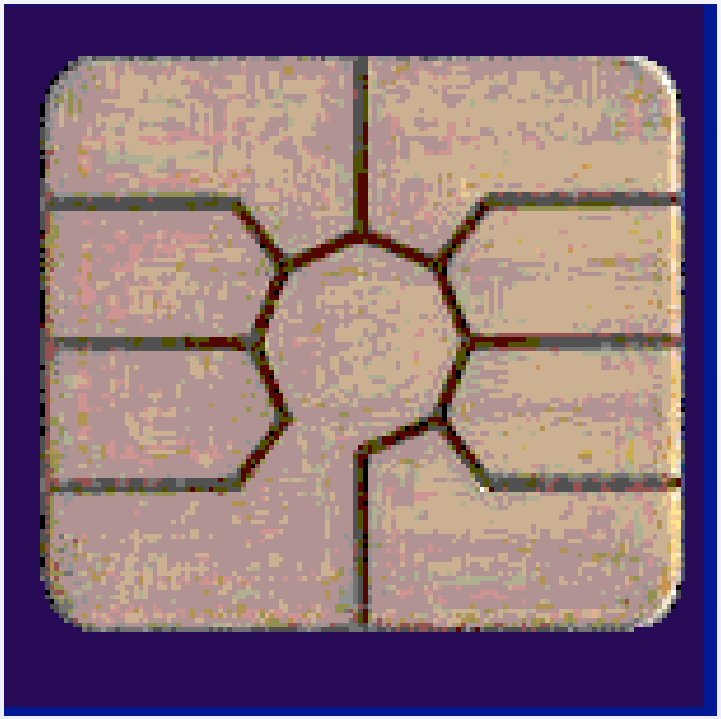
- La Banca invia un messaggio casuale (un numero casuale) alla carta attraverso un terminale bancomat;
- La carta usa la chiave privata dell’utente (inserita in essa) per codificare tale numero casuale e invia il risultato alla banca tramite il terminale bancomat.

Come funziona una carta bancomat “sicura”/3

La banca accetta questa come prova dell’identità dell’utente perché la può verificare dato che tutti, inclusa la banca, conoscono la chiave pubblica ma solo l’utente conosce la propria chiave segreta (e quindi nessun altro può spacciarsi per lui).
Inoltre i messaggi scambiati con la banca NON possono rivelare la chiave privata dell’utente perché bisognerebbe saper risolvere un problema matematico computazionalemente “difficile” (se il sistema è stato sviluppato correttamente)

Come funziona una carta bancomat “sicura”/4

Per fare ciò serve una carta “intelligente” (*smart-card*) cioè in grado di svolgere i calcoli necessari e che non possa essere manomessa senza provocarne il mancato funzionamento; sono le carte che contengono un chip come questo



Un attacco a RSA di tipo chosen-ciphertext

Un intruso desidera determinare il testo in chiaro M di una codifica C inviata ad A:

- sceglie casualmente un intero R e richiedere ad A la decodifica di $C_1 \equiv R^{e_A} C \bmod n_A$.

Un attacco a RSA di tipo chosen-ciphertext

Un intruso desidera determinare il testo in chiaro M di una codifica C inviata ad A:

- sceglie casualmente un intero R e richiedere ad A la decodifica di $C_1 \equiv R^{e_A} C \bmod n_A$.
- Se A esegue (non ha possibilità di accorgersi della frode perché, a causa di R, C_1 non ha apparentemente nulla a che spartire con C), l'intruso ottiene quindi la quantità $M_1 \equiv C_1^{d_A} \bmod n_A$.

Un attacco a RSA di tipo chosen-ciphertext

Un intruso desidera determinare il testo in chiaro M di una codifica C inviata ad A:

- sceglie casualmente un intero R e richiedere ad A la decodifica di $C_1 \equiv R^{e_A} C \bmod n_A$.
- Se A esegue (non ha possibilità di accorgersi della frode perché, a causa di R, C_1 non ha apparentemente nulla a che spartire con C), l'intruso ottiene quindi la quantità $M_1 \equiv C_1^{d_A} \bmod n_A$.
- La decodifica di C segue calcolando $M_1 R^{-1} \equiv R^{e_A d_A} C^{d_A} R^{-1} \equiv C^{d_A} \equiv M \mod n_A$.

Un attacco a RSA di tipo chosen-ciphertext

Un intruso desidera determinare il testo in chiaro M di una codifica C inviata ad A:

- sceglie casualmente un intero R e richiedere ad A la decodifica di $C_1 \equiv R^{e_A} C \bmod n_A$.
- Se A esegue (non ha possibilità di accorgersi della frode perché, a causa di R, C_1 non ha apparentemente nulla a che spartire con C), l'intruso ottiene quindi la quantità $M_1 \equiv C_1^{d_A} \bmod n_A$.
- La decodifica di C segue calcolando $M_1 R^{-1} \equiv R^{e_A d_A} C^{d_A} R^{-1} \equiv C^{d_A} \equiv M \mod n_A$.

In conclusione: mai applicare la propria funzione di decifratura (o la propria firma digitale) ad un documento casuale; è opportuno utilizzare una funzione *hash* prima di firmare digitalmente.

Attacchi elementari

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)
- scelta del modulo n fissata per tutti gli utenti.

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)
- scelta del modulo n fissata per tutti gli utenti.
  - se un utente A sa che B ha lo stesso suo n può facilmente calcolare d_B

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)
- scelta del modulo n fissata per tutti gli utenti.
  - se un utente A sa che B ha lo stesso suo n può facilmente calcolare d_B
  - se C_1, C_2 sono due codifiche di un messaggio M, mediante due chiavi pubbliche (n, e_1), (n, e_2), dove $(e_1, e_2) = 1$, calcolare la relazione

    $$(C_1^{-1})^{-r} C_2^s \equiv M^{re_1+se_2} = M \bmod n.$$

    con $r, s \in \mathbb{Z}$ tali che $re_1 + se_2 = 1$.

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)
- scelta del modulo n fissata per tutti gli utenti.
  - se un utente A sa che B ha lo stesso suo n può facilmente calcolare d_B
  - se C_1, C_2 sono due codifiche di un messaggio M, mediante due chiavi pubbliche (n, e_1), (n, e_2), dove $(e_1, e_2) = 1$, calcolare la relazione
    $$(C_1^{-1})^{-r} C_2^s \equiv M^{re_1+se_2} = M \bmod n.$$
    con $r, s \in \mathbb{Z}$ tali che $re_1 + se_2 = 1$.
    Si noti che la condizione $(e_1, e_2) = 1$ è molto spesso verificata in pratica.

Attacchi elementari

- Scelta dei possibili messaggi in chiaro troppo limitata (che consente un attacco di tipo ricerca esaustiva: si provano a codificare tutti i messaggi in chiaro fino a che non viene determinato quello la cui codifica è uguale al messaggio intercettato)
- scelta del modulo n fissata per tutti gli utenti.
  - se un utente A sa che B ha lo stesso suo n può facilmente calcolare d_B
  - se C_1, C_2 sono due codifiche di un messaggio M, mediante due chiavi pubbliche (n, e_1), (n, e_2), dove $(e_1, e_2) = 1$, calcolare la relazione
    $$(C_1^{-1})^{-r} C_2^s \equiv M^{re_1+se_2} = M \bmod n.$$
    con $r, s \in \mathbb{Z}$ tali che $re_1 + se_2 = 1$.
    Si noti che la condizione $(e_1, e_2) = 1$ è molto spesso verificata in pratica.

In conclusione: ogni n non deve poter essere usato da più di un utente.

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$.

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$. Se (n, e) è la chiave pubblica e $(M, n) = 1$, allora M è un punto fisso per f se e solo se $o_p(M) \mid (e-1)$ e $o_q(M) \mid (e-1)$,

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$.
Se (n, e) è la chiave pubblica e $(M, n) = 1$, allora M è un punto fisso per f se e solo se $o_p(M) \mid (e-1)$ e $o_q(M) \mid (e-1)$,
Allora

$$\text{card}\{M \in \mathbb{Z}_p^* \ : \ o_p(M) \mid (e-1)\} = \sum_{\substack{d \mid (e-1) \\ d \mid (p-1)}} \varphi(d)$$

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$.
Se (n, e) è la chiave pubblica e $(M, n) = 1$, allora M è un punto fisso per f se e solo se $o_p(M) \mid (e-1)$ e $o_q(M) \mid (e-1)$,
Allora

$$\text{card}\{M \in \mathbb{Z}_p^* \ : \ o_p(M) \mid (e-1)\} = \sum_{\substack{d \mid (e-1) \\ d \mid (p-1)}} \varphi(d)$$

$$= \sum_{d \mid (e-1, p-1)} \varphi(d) = (e-1, p-1)$$

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$.
Se (n, e) è la chiave pubblica e $(M, n) = 1$, allora M è un punto fisso per f se e solo se $o_p(M) \mid (e-1)$ e $o_q(M) \mid (e-1)$,
Allora

$$\text{card}\{M \in \mathbb{Z}_p^* \ : \ o_p(M) \mid (e-1)\} = \sum_{\substack{d \mid (e-1) \\ d \mid (p-1)}} \varphi(d)$$

$$= \sum_{d \mid (e-1, p-1)} \varphi(d) = (e-1, p-1)$$

e quindi

$$\text{card}\{M \in \mathbb{Z}_n^* \ ; \ M^e \equiv M \bmod n\} = (e-1, p-1) \cdot (e-1, q-1).$$

Punti fissi

Se M è punto fisso della funzione di cifratura si ha $f(M) \equiv M \bmod n$.
Se (n, e) è la chiave pubblica e $(M, n) = 1$, allora M è un punto fisso per f se e solo se $o_p(M) \mid (e-1)$ e $o_q(M) \mid (e-1)$,
Allora

$$\text{card}\{M \in \mathbb{Z}_p^* \ : \ o_p(M) \mid (e-1)\} = \sum_{\substack{d \mid (e-1) \\ d \mid (p-1)}} \varphi(d)$$

$$= \sum_{d \mid (e-1, p-1)} \varphi(d) = (e-1, p-1)$$

e quindi

$$\text{card}\{M \in \mathbb{Z}_n^* \ ; \ M^e \equiv M \bmod n\} = (e-1, p-1) \cdot (e-1, q-1).$$

In conclusione: e deve essere scelto in modo tale che $(e-1, p-1) \cdot (e-1, q-1)$ sia piccolo (altrimenti si hanno molti punti fissi).

Broadcast Attack (e “piccolo”)

A manda lo stesso messaggio M a più destinatari $B_1, \ldots, B_r$ aventi chiavi pubbliche (n_i, e), $i = 1, \ldots, r$.

$$(n_i, n_j) = 1 \text{ per } i \neq j \quad \text{e} \quad M < n = \min(n_i)$$

Broadcast Attack (e “piccolo”)

A manda lo stesso messaggio M a più destinatari $B_1, \ldots, B_r$ aventi chiavi pubbliche (n_i, e), $i = 1, \ldots, r$.

$$(n_i, n_j) = 1 \text{ per } i \neq j \quad \text{e} \quad M < n = \min(n_i)$$

Un intruso intercetta r messaggi cifrati $C_i \equiv M^e \bmod n_i$.

Broadcast Attack (e "piccolo")

A manda lo stesso messaggio M a più destinatari $B_1, \ldots, B_r$ aventi chiavi pubbliche (n_i, e), $i = 1, \ldots, r$.

$$(n_i, n_j) = 1 \text{ per } i \neq j \quad \text{e} \quad M < n = \min(n_i)$$

Un intruso intercetta r messaggi cifrati $C_i \equiv M^e \bmod n_i$.
Se $r \geq e$, allora l'intruso ne sceglie e e, grazie al Teorema Cinese del Resto, risolve il sistema $C \equiv C_i \bmod n_i$, $i = 1, \ldots, e$, ottenendo un'unica soluzione C definita modulo $\prod_{i=1}^{e} n_i$.

Broadcast Attack (e "piccolo")

A manda lo stesso messaggio M a più destinatari $B_1, \ldots, B_r$ aventi chiavi pubbliche (n_i, e), $i = 1, \ldots, r$.

$$(n_i, n_j) = 1 \text{ per } i \neq j \quad \text{e} \quad M < n = \min(n_i)$$

Un intruso intercetta r messaggi cifrati $C_i \equiv M^e \bmod n_i$.
Se $r \geq e$, allora l'intruso ne sceglie e e, grazie al Teorema Cinese del Resto, risolve il sistema $C \equiv C_i \bmod n_i$, $i = 1, \ldots, e$, ottenendo un'unica soluzione C definita modulo $\prod_{i=1}^{e} n_i$.

Ma $M < n_i$ e quindi $M^e < \prod_{i=1}^{e} n_i$ da cui segue $M^e = C$.

Broadcast Attack (e "piccolo")

A manda lo stesso messaggio M a più destinatari $B_1, \ldots, B_r$ aventi chiavi pubbliche (n_i, e), $i = 1, \ldots, r$.

$$(n_i, n_j) = 1 \text{ per } i \neq j \quad \text{e} \quad M < n = \min(n_i)$$

Un intruso intercetta r messaggi cifrati $C_i \equiv M^e \bmod n_i$.
Se $r \geq e$, allora l'intruso ne sceglie e e, grazie al Teorema Cinese del Resto, risolve il sistema $C \equiv C_i \bmod n_i$, $i = 1, \ldots, e$, ottenendo un'unica soluzione C definita modulo $\prod_{i=1}^{e} n_i$.

Ma $M < n_i$ e quindi $M^e < \prod_{i=1}^{e} n_i$ da cui segue $M^e = C$.
Per ottenere M è dunque sufficiente calcolare la radice e-esima intera di C (estrarre radici modulo n è difficile, ma estrarre radici intere è computazionalmente facile)

Random Faults, I

È possibile eseguire l'operazione di esponenziazione modulare utilizzando il Teorema Cinese del Resto (riduce costo computazionale di un fattore quattro). Ma espone RSA al seguente attacco.

Random Faults, I

È possibile eseguire l'operazione di esponenziazione modulare utilizzando il Teorema Cinese del Resto (riduce costo computazionale di un fattore quattro). Ma espone RSA al seguente attacco.

- A firma il proprio messaggio M inviando a B la coppia $(f_B(M), f_B(f_A^{-1}(M)))$.

Random Faults, I

È possibile eseguire l'operazione di esponenziazione modulare utilizzando il Teorema Cinese del Resto (riduce costo computazionale di un fattore quattro). Ma espone RSA al seguente attacco.

- A firma il proprio messaggio M inviando a B la coppia $(f_B(M), f_B(f_A^{-1}(M)))$.
- Durante il calcolo di $f_A^{-1}(M) \equiv M^{d_A} \bmod n_A$ (fatto con il Teorema Cinese del Resto) avviene un errore casuale che modifica un bit del risultato.

Random Faults, I

È possibile eseguire l'operazione di esponenziazione modulare utilizzando il Teorema Cinese del Resto (riduce costo computazionale di un fattore quattro). Ma espone RSA al seguente attacco.

- A firma il proprio messaggio M inviando a B la coppia $(f_B(M), f_B(f_A^{-1}(M)))$.
- Durante il calcolo di $f_A^{-1}(M) \equiv M^{d_A} \bmod n_A$ (fatto con il Teorema Cinese del Resto) avviene un errore casuale che modifica un bit del risultato.
- Sia $C_p \equiv M^{d_A} \bmod p_A$ corretto e $\widehat{C}_q \equiv M^{d_A} \bmod q_A$ non corretto.

Random Faults, I

È possibile eseguire l'operazione di esponenziazione modulare utilizzando il Teorema Cinese del Resto (riduce costo computazionale di un fattore quattro). Ma espone RSA al seguente attacco.

- A firma il proprio messaggio M inviando a B la coppia $(f_B(M), f_B(f_A^{-1}(M)))$.
- Durante il calcolo di $f_A^{-1}(M) \equiv M^{d_A} \bmod n_A$ (fatto con il Teorema Cinese del Resto) avviene un errore casuale che modifica un bit del risultato.
- Sia $C_p \equiv M^{d_A} \bmod p_A$ corretto e $\widehat{C}_q \equiv M^{d_A} \bmod q_A$ non corretto.
- B trova, dalla seconda parte, la quantità

$$\widehat{C}_A \equiv t_1 C_p + t_2 \widehat{C}_q \bmod n_A,$$

dove $t_1 \equiv 1 \bmod p_A$, $t_1 \equiv 0 \bmod q_A$, $t_2 \equiv 1 \bmod q_A$, $t_2 \equiv 0 \bmod p_A$.

Random Faults, II

- B si accorge quindi che la firma non è corretta perché $(\widehat{C}_A)^{e_A} \not\equiv M \bmod n_A$.

Random Faults, II

- B si accorge quindi che la firma non è corretta perché $(\widehat{C}_A)^{e_A} \not\equiv M \bmod n_A$.
- B prova a ridurre modulo p_A e q_A ottenendo $(\widehat{C}_A)^{e_A} \equiv M \bmod p_A$ e $(\widehat{C}_A)^{e_A} \not\equiv M \bmod q_A$.

Random Faults, II

- B si accorge quindi che la firma non è corretta perché $(\widehat{C}_A)^{e_A} \not\equiv M \bmod n_A$.
- B prova a ridurre modulo p_A e q_A ottenendo $(\widehat{C}_A)^{e_A} \equiv M \bmod p_A$ e $(\widehat{C}_A)^{e_A} \not\equiv M \bmod q_A$.
- B, calcolando $(n_A, (\widehat{C}_A)^{e_A} - M)$, ottiene la fattorizzazione di n_A.

Random Faults, II

- B si accorge quindi che la firma non è corretta perché $(\widehat{C}_A)^{e_A} \not\equiv M \bmod n_A$.
- B prova a ridurre modulo p_A e q_A ottenendo $(\widehat{C}_A)^{e_A} \equiv M \bmod p_A$ e $(\widehat{C}_A)^{e_A} \not\equiv M \bmod q_A$.
- B, calcolando $(n_A, (\widehat{C}_A)^{e_A} - M)$, ottiene la fattorizzazione di n_A.

È quindi necessario che A verifichi la correttezza del calcolo della propria firma digitale prima di inviarla a B. Inoltre, nel caso A inserisca una modificazione casuale del messaggio prima di effettuare la firma, questo tipo di attacco fallisce.