ATTACHI all’ RSA

Uno sguardo d'assieme

- La fattorizzazione
- Forzare RSA
- Sicurezza RSA

Saveri Daniele

Fattorizzazione

- **Def**: Fattorizzare (ridurre in fattori) un numero n significa trovare un insieme di numeri {a0, a1, a2, a3...} tali che il loro prodotto sia il numero originario (n = a0 * a1 * a3 * a3...).
- **Def**: Se un numero ha come proprio unico fattore se stesso, è un numero primo (Ad es: 5=51)
- Importante:ogni numero naturale ha una ed una sola fattorizzazione in numeri primi.

Esempio

52=2^2 x 13^1 oppure 3300=2^2_3^1_5^2_11^1

Algoritmi di fattorizzazione

- I migliori algoritmi di fattorizzazione noti non sono polinomiali, ma *subesponenziali*
- *Il tempo T(n) che impiegano (nel caso peggiore in cui N sia il prodotto di due primi di uguale lunghezza) è proporzionale a* $e^{\sqrt{\log(N)*\log(\log(N))}}$
- Gli algoritmi capaci di fattorizzare in modo efficiente numeri molto grandi, sono due:
  1. ***Quadratic Sieve***
  2. ***Number Field Sieve (NFS)***
- Altri algoritmi *:*

  Elliptic Curve Method (ECM), ***Trial Division*** e **Algoritmo di *Lehmann***

Trial Division

- Più vecchio algoritmo di fattorizzazione
- Controlla ogni primo minore o uguale della radice quadrata del numero da fattorizzare
- Svantaggio: richiede fino a $\sqrt{n}$ operazioni per scomporre numeri che hanno fattori primi molto vicini fra loro

Esempio

N1= 3992003 = 1997 x 1999

N2=122222221 = 111111111 x 11

Algoritmo di Lehman

- Se riusciamo a trovare x e y tali che
  $N + y^2 = x^2$ allora
  $N = x^2 - y^2 = (x+y)*(x-y)$
- N è scomposto in due fattori

Esempio precedente se y=1 troviamo:

$N+1=1998^2$ e quindi
$N=(1998-1) * (1998+1)=1997 * 1999$

- Problema:quale dei due è più efficiente?

Soluzione: si mescolano i due algoritmi

Algoritmo di Lehman

- Passo 1: $2<m<R=N^{(1/3)}$
  si applica trial division con m dispari
  Se qualche divisione è esatta l'algoritmo termina
  ALTRIMENTI
- Passo 2: si pone $k=1$
  $x_0=[\sqrt{4kN}]$, $x_1=[\sqrt{4kN}+\sqrt{N/k}/4R]$
- Si verifica se c'è un x cui il valore di
  $x^2 - 4KN$ è un quadrato perfetto
  se così l'algoritmo termina
  con il calcolo del $\gcd(x+y,N)$
- Altrimenti si ripete il passo 2 aumentando k

Esempio

- N=2881 allora R=14
- N non ha fattori primi <14
- Utilizzando il metodo descritto si ha
  k=1 x=110
- 110^2 – 4* 2881 =24^2
- Da cui deduciamo con il calcolo del gcd p=43 e q=67

Quadratic sieve

- Algoritmo più veloce conosciuto per numeri con meno di 150 cifre
- Inventato da Pomerance agli inizi degli anni ‘80
- Progettò una macchina di fattorizzazione modulare : la grandezza del numero da fattorizzare era direttamente proporzionale alla grandezza della macchina

Quadratic sieve: l'idea(1)

- Dato il numero da fattorizzare n:

  trovare x e y tali che

  x^2 _ y^2 mod n

  x ≠ ± y mod n
- Il fattore di n sarà gcd(x-y,n).
- Uso di una *factor base*, un insieme di numeri primi piccoli

Quadratic sieve: l'idea(2)

- Si ricavano diversi interi x tali che i fattori primi di x^2 mod n stiano nella factor base
- Si prendono i prodotti di diversi x in maniera tale che tutti i fattori primi di x^2 mod n vengano utilizzati un numero pari di volte
- Avremo una congruenza del tipo

  x^2 _ y^2 mod n

  che ci porta alla fattorizzazione di n

Quadratic sieve: Esempio

- Consideriamo *n = 15770708441*.
- Sia *B = {2; 3; 5; 7; 11; 13}* la *factor base*.
- Consideriamo le tre congruenze:

  (8340934156)_ _ 3 _ 7 mod n

  (12044942944)_ _ 2 _ 7 _ 13 mod n

  (2773700011)_ _ 2 _ 3 _ 13 mod n
- Se prendiamo il prodotto di queste tre congruenze, abbiamo:

  *(8340934156 _ 12044942944 _ 2773700011)_*

  _ (2 _ 3 _ 7 _ 13) _ mod n

…continua

- Riducendo l'espressione nella parentesi modulo n, abbiamo

  (9503435785)_ _ (546)_ mod n

- Quindi calcolando:

  gcd(9503435785-546; 15770708441); troviamo il fattore *115759* di *n*.

Attaccare RSA

- Conoscendo la chiave pubblica (n,e) l'attaccante vuole calcolare la chiave privata:

  d _ e^-1 mod _(n)

  _(n)=(p-1)(q-1)

- Possibile solo se riesce a fattorizzare _(n)

Attaccare RSA

- Forzare RSA equivale a calcolare _(n)
- Se n e _(n) sono conosciuti

  n=p*q (con p e q numeri primi)

  _(n)=(p-1)(q-1)
- Sostituendo q=n/p nella seconda, otteniamo

  p^2 – (n- _(n) + 1) * p + n = 0

Le due radici di questa equazione sono p e q

Esempio

- Supponiamo che un attaccante conosca il valore di _(n) e n

  _(n)=84754668 e n=84773093

- Queste informazioni permettono di scrivere l'equazione

  p^2 – 18426p + 84773093 = 0

- Risolvendo si ottengono le radici:9539 e 8887 che sono i fattori p e q di n

Attaccare RSA

- Conoscendo la chiave pubblica (n,e) e il messaggio cifrato C = M^e mod n

  l'attaccante vuole risalire a M
- Dato C potrebbero esserci informazioni parziali sul messaggio M facili da ottenere (senza dover decifrare C)
- Problema ancora aperto perché non si sa se la computazione di M ha complessità pari o superiore alla fattorizzazione

Sicurezza RSA

- Sicurezza assoluta data la difficoltà nel fattorizzare un numero estremamente elevato
- Solo conoscendo la fattorizzazione di *n* è possibile trovare il valore delle chiavi
- La funzione di cifratura *x^e mod n* è una funzione "one-way"

<u>Esempio</u>

N*= 664 bit fattorizzabile con 10^23 passi*

rete*: un milione di computer ,ciascuno esegue un milione di passi al secondo*

tempo impiegato *pari circa a 4000 anni*

Stima dello sforzo richiesto

Dimensione della chiave in bits	Anni – MIPS necessari per la fattorizzazione
512	30.000
768	200.000.000
1024	300.000.000.000
2048	300.000.000.000.000.000.000

Sicurezza RSA

- Difficoltà nel fattorizzare n
- Per garantire l'inviolabilità bisogna rendere la fattorizzazione un'operazione computazionalmente irrealizzabile
- Si usano chiavi di 1024 e 2048 bits.

La sfida dell'RSA Security

- Vengono premiati coloro che fattorizzano gli interi proposti da loro in una particolare lista.
- Il 3 dicembre del 2003, è stato fattorizzato rsa576:

  un numero di 576 bit, 174 cifre decimali, prodotto di due primi di 87 cifre

Argomenti trattati

- Scelta dei parametri *p* e *q*
- RSA randomizzato
- Cycling attack

Elena Pugliesi

Scelta di *p* e *q*

La scelta dei parametri *p* e *q* dovrebbe essere fatta con cura.

Ad esempio, *p* e *q* non dovrebbero essere troppo vicini:

- Supponiamo che sia $p > q$; se *p* e *q* fossero vicini, allora:
  - Il valore $(p - q) / 2$ sarebbe piccolo
  - Il valore $(p + q) / 2$ sarebbe solo leggermente più grande di $\sqrt{n}$

Scelta di *p* e *q*

- Vale sempre l'uguaglianza:

$$(p+q)^2/4 - n = (p-q)^2/4$$

Da cui si deduce che $(p+q)^2/4 - n$ è un quadrato perfetto. Per fattorizzare *n* basterà testare gli interi $x > \sqrt{n}$ finchè non se ne trova uno per cui il valore di $x^2 - n$ è un quadrato perfetto che chiameremo y^2.
Abbiamo:

$$x^2 - n = y^2$$

Da cui:

$$n = x^2 - y^2 = (x+y)(x-y)$$

Quindi $p = x + y$ e $q = x - y$.

Scelta di *p* e *q*

Anche il valore di _*(n)* dovrebbe essere tenuto in considerazione nella scelta di *p* e di *q*.

- Supponiamo che (*p* - 1, *q* - 1) sia grande e, di conseguenza, che il minimo comune multiplo *u* di *p* - 1 e *q* – 1 sia piccolo rispetto a _*(n)*.
- Allora, un qualsiasi inverso di *e* modulo *u* potrà essere usato come esponente di decifratura (al posto di *d*; questo fatto deriva dal Teorema di Eulero).
- dato che *u* è relativamente piccolo, un tale inverso di *e* potrebbe essere trovato per tentativi.

Pertanto, è meglio che *p* - 1 e *q* - 1 non abbiano alcun fattore comune grande.

Teorema sulla computazione di *d*

TEOREMA:

Un algoritmo per computare d può essere convertito in un algoritmo probabilistico per fattorizzare n.

- Questo teorema da un'indicazione della difficoltà presunta di ricavare la chiave segreta d a partire da quella pubblica $(n; e)$: se fossimo in grado di risolvere questo problema, allora saremmo anche in grado di esibire un algoritmo efficiente per la fattorizzazione.

Teorema sulla computazione di *d*

Il precedente teorema, insieme al fatto che il problema della fattorizzazione viene ritenuto intrattabile, giustica il fatto che RSA sia in generale ritenuto un crittosistema sicuro.

- Gli attacchi che gli sono stati sferrati fin dalla sua nascita sono molti, ma tutti funzionano solo in particolari condizioni e sono quindi facilmente evitabili scegliendo i parametri (p; q; e; d; n) in maniera opportuna.

RSA randomizzato

Supponiamo che Alice e Bob comunichino utilizzando RSA, e che molto spesso Alice debba inviare a Bob un singolo bit confidenziale: $b \in \{0,1\}$.

- Problema:
  Alice dovrebbe cifrare questo bit *b* come se fosse un messaggio normale, cioè calcolando $b^e \bmod n$?

RSA randomizzato

- Risposta:
  Ovviamente no. Dato che $b^e = b$ per $b \in \{0,1\}$, il testo cifrato sarebbe identico al testo in chiaro.
- Cosa potrebbe fare Alice per superare questo inconveniente?
  - Potrebbe generare un intero *x* a caso tale che *x* < *n*/2 e trasmettere a Bob $y = (2x + b)^e \bmod n$
  - Ricevuto *y*, Bob userebbe la propria chiave privata per recuperare 2*x* + *b*; a questo punto, *b* sarebbe il bit meno significativo di tale numero intero.

RSA randomizzato

- Questo metodo di trasmettersi i dati è sicuro?
  - Può essere al più sicuro quanto lo è RSA, dato che se un avversario potesse recuperare $2x + b$ a partire da y allora potrebbe anche determinare il valore di b.

- E se fosse possibile recuperare b a partire da y e da e, senza necessariamente riuscire a recuperare tutto il testo in chiaro $2x + b$?
  - si può dimostrare che il metodo esposto per cifrare un bit nel bit meno significativo di un intero è sicuro esattamente quanto RSA

RSA randomizzato

Quindi ogni metodo che sia in grado di indovinare il valore di *b* da $(2x+b)^e \bmod n$ e da *e* con probabilità di successo significativamente maggiore di 1/2 può essere utilizzato per rompere il crittosistema RSA.

Questo fatto e la considerazione che il messaggio non è altro che una sequenza di bit apre una possibilità interessante.

RSA randomizzato

- Alice potrebbe allora inviare un messaggio qualsiasi a Bob suddividendolo in bit e trasmettendo un bit alla volta, scegliendo ogni volta in maniera indipendente un *x* a caso tra 0 e $\left\lfloor \frac{n}{2} \right\rfloor - 1$.

Il crittosistema randomizzato che ne risulta è chiaramente molto sicuro: l'analisi del testo cifrato risulta essere molto più difficile di quanto avviene con l'RSA tradizionale.

RSA randomizzato

- D'altra parte, dobbiamo inviare un blocco cifrato per ogni bit del messaggio in chiaro:
  - Questo significa che se il messaggio da trasmettere è molto lungo sarà necessaria una velocità di trasmissione molto elevata oppure un intervallo di tempo molto lungo.

Attacchi attivi e passivi

- **Attacchi attivi e passivi**: a seconda che l'attaccante sia in grado o meno di inviare alla cifratura testi in chiaro di propria scelta.

  Il più semplice tipo di attacco è quello di **forza bruta** ovvero provare tutte le chiavi sino a trovare quella che decifra correttamente il messaggio.

Attacchi attivi e passivi

- **Attacchi attivi** : intervengono su altri algoritmi di crittografia a chiave pubblica (es. man in the middle).
- **Attacchi passivi**: sono computazionalmente inefficienti.
  - Il loro scopo è quello di calcolare *i* tale che $b = a^i \bmod p$ dati *a*, *b* e *p.*
  - Sono facilitati se *p* non è un numero primo sicuro. Ossia se *((p – 1) / 2)* non è primo.
  - Come contrattacco è possibile scegliere *n* con il numero maggiore di bit possibile ma questo aumenterebbe la lentezza dell’algoritmo.

Attacchi matematici e teorici

- **Attacchi matematici**: la fattorizzazione. Gli algoritmi utilizzati in questo caso sono inefficienti.
  - Il problema della fattorizzazione viene ritenuto intrattabile.
- **Attacchi teorici alle implementazioni**:
  - Common modulus
  - Low exponent
  - Chosen ciphertext

Cycling Attacks

Dato un testo cifrato *c* tale che:

$$c = m^e \bmod n$$

e un intero positivo *k* tale che:

$$c^{e^k} \equiv c \pmod{n}$$

Poiché la criptazione è una permutazione nello spazio {0, 1, …., n-1} dovrà esistere un intero *k* definito come sopra.

Per lo stesso motivo esisterà il caso in cui:

$$c^{e^{k-1}} \equiv m \pmod{n}$$

Queste osservazioni ci conducono al seguente Cycling attack sull'RSA.

Cycling Attacks

Un attaccante computa:

$$c^{e} \bmod n, c^{e^{2}} \bmod n, c^{e^{3}} \bmod n, \ldots.$$

Finchè non ottiene *c* per la prima volta.

Se:

$$c^{e^{k}} \bmod n = c$$

allora il numero che lo precede all'interno del ciclo, che chiameremo

$$c^{e^{k-1}} \bmod n$$

corrisponderà al testo in chiaro *m*.

Cycling Attacks

Un cycling attack generalizzato consiste nel trovare il più piccolo intero positivo *u* tale che:

$$f = \gcd(c^{e^u} - c; n) > 1$$

- Se:

  $c^{e^u} \equiv c \pmod{p}$ e $c^{e^u} \not\equiv c \pmod{q}$

  allora $f = p$.
- Se:

  $c^{e^u} \not\equiv c \pmod{p}$ e $c^{e^u} \equiv c \pmod{q}$

  allora $f = q$.

In entrambi i casi, *n* è stato fattorizzato, e l'avversario può ricavare *d* e quindi anche *m*.

Cycling Attacks

- Se:

$$c^{e^u} \equiv c(\bmod\, p) \quad \text{e} \quad c^{e^u} \equiv c(\bmod\, q)$$

si verifica che $f = n$ e $c^{e^u} \equiv c(\bmod\, n)$.

Infatti, u coincide con il più piccolo intero positivo k per cui vale che:

$$c^{e^k} \equiv c(\bmod\, n)$$

In questo caso il cycling attack ha avuto successo e quindi $m = c^{e^{u-1}} \bmod n$ può essere computata efficientemente.

Cycling Attacks

Poiché il caso in cui:

$$c^{e^u} \equiv c \pmod{p} \quad \text{e} \quad c^{e^u} \equiv c \pmod{q}$$

si verifica molto meno frequentemente dei casi:

$$c^{e^u} \equiv c \pmod{p} \quad \text{e} \quad c^{e^u} \not\equiv c \pmod{q}$$

$$c^{e^u} \not\equiv c \pmod{p} \quad \text{e} \quad c^{e^u} \equiv c \pmod{q}$$

il cycling attack generalizzato termina prima del termine del cycling attack.

Cycling Attacks

Per questa ragione il cycling attack generalizzato può essere essenzialmente visto come un algoritmo per fattorizzare *n*.

Questo ci da la possibilità di affermare che il cycling attack non è una minaccia per la sicurezza dell’RSA.

Attacchi alla implementazione

- Common modulus
- Low exponent
- Chosen chipertext

Zanobi Riccardo

Attacco Common Modulus

- Vulnerabilità di RSA se viene utilizzato lo stesso modulo da due utenti.

Aldo (1)	**Bruno (2)**
Chiave pubblica $e_1 \quad n$ Chiave privata $d_1 \quad n$	Chiave pubblica $e_2 \quad n$ Chiave privata $d_2 \quad n$

- Sia $mcd(e_1,e_2)=1$

Attacco Common Modulus

- Viene inviato un messaggio m ad entrambi.

Aldo | **Bruno**

$$c_1 = m^{e_1} \bmod n \qquad\qquad c_2 = m^{e_2} \bmod n$$

- Marco intercetta i due messaggi, e tramite l'algoritmo di Euclide può calcolare due numeri,r ed s tali che:

$$re_1 + se_2 = 1$$

Attacco Common Modulus

- Supponiamo r negativo (sono sempre uno positivo e uno negativo),se possiamo calcolare:

$$c_1^{-1} \quad \Longrightarrow \quad c_1 x \equiv 1 \bmod n$$

- Possiamo risalire al messaggio originale, tramite le seguenti uguaglianze:

$$(c_1^{-1})^{-r} c_2^{s} = c_1^{r} c_2^{s} = (m^{re_1})(m^{se_2}) \bmod n =$$
$$(m^{(re_1+se_2)}) \bmod n = m$$

Attacco Common Modulus

- Esempio con n=33, spediamo il messaggio m=20
  - Scegliamo due esponenti diversi: $e_1 = 7, e_2 = 19$
  - e_1 ed e_2 sono coprimi tra loro
  - Marco intercetta i due messaggi cifrati

$$c_1 = 20^7 \bmod 33 = 26$$

$$c_2 = 20^{19} \bmod 33 = 5$$

  - 26 è coprimo con 33 quindi è invertibile

Attacco Common Modulus

- Esempio con n=33, spediamo il messaggio m=20
  - Marco calcola tramite l'algoritmo di Euclide esteso, i numeri r ed s.

Euclide-Esteso(a,b)
if(b=0) then
return (a,1,0)
else

$$(d', x', y') \leftarrow Euclide-Esteso(b, a \bmod b)$$
$$(d, x, y) \leftarrow (d', y', x' - \lfloor a/b \rfloor y')$$
$$return(d, x, y)$$

  - r=-8, s=3: infatti $-8 \cdot 7 + 3 \cdot 19 = -56 + 57 = 1$

Attacco Common Modulus

- Esempio con n=33, spediamo il messaggio m=20
  - calcoliamo l'inverso di c_1

$$26x \equiv 1 \bmod 33$$

$$x = 14$$

  - ed eseguiamo un po' di conti…

$$14^8 \cdot 5^3 \bmod 33 = 16 \cdot 26 \bmod 33 = 20$$

Attacco Low Exponent

- Il valore dell'esponente e può essere arbitrario, anche se è consigliabile scegliere un valore uguale per tutti gli utenti del sistema.
- Vantaggio di un e piccolo: maggiore velocità nella fase di cifratura e di firma.

Attacco Low Exponent

- Scelta di e = 2:
  - Impossibile perché non può essere coprimo con (p-1)(q-1)
  - Ci sono comunque interessi teorici in quanto modificando RSA e scegliendo e=2, un attacco di tipo ciphertext-only diventerebbe difficile quanto fattorizzare (Rabin,Williams)

Attacco Low Exponent

- Scelta di e = 3:
  - Tre utenti scelgono e uguale ma moduli diversi (supponiamo siano a due a due coprimi)

$$e_A = e_B = e_C = 3 \qquad n_A, n_B, n_C$$

  - Un quarto utente vuole mandare lo stesso messaggio m ad A,B,C

$$c_A = m^3 \bmod n_A$$

$$c_B = m^3 \bmod n_B$$

$$c_C = m^3 \bmod n_C$$

Attacco Low Exponent

- Scelta di e = 3:
  - Chiunque intercetti i tre messaggi cifrati può risalire ad m.
  - E'possibile calcolare x con il Teorema Cinese del resto

$$\begin{cases} x \equiv c_A \bmod n_A \\ x \equiv c_B \bmod n_B \\ x \equiv c_C \bmod n_C \end{cases}$$

Attacco Low Exponent

- Il teorema cinese del resto:
  - Dato un sistema di equazioni modulari

$$\begin{cases} x \equiv a_1 \bmod n_1 \\ x \equiv a_2 \bmod n_2 \\ \vdots \\ x \equiv a_r \bmod n_r \end{cases}$$

  - Con gli n_i primi fra loro due a due
  - Siano:

$$N = \prod_{i=1}^{r} n_i \qquad N_i = \frac{N}{m_i} \qquad Y_i = N_i^{-1} \bmod n_i$$

  - La soluzione generale del sistema è:

$$x = \sum_{i=1}^{r} a_i N_i Y_i \bmod N$$

Attacco Low Exponent

- Scelta di e = 3:
  - La soluzione è unica a meno di multipli di N (con N uguale al prodotto dei tre moduli), quindi esiste una soluzione x* compresa tra 0 e N-1
  - Poiché

    $$m^3 < N$$

    Risulta $x^* = m^3$
  - Basta calcolare la radice cubica di x* per ricavare m

Attacco Low Exponent

- Esempio con m=13:
  - Prendiamo tre moduli diversi

$$n_A = 15, n_B = 77, n_C = 221$$

  - Quindi abbiamo:

$$c_A = 13^3 \bmod 15 = 7$$

$$c_B = 13^3 \bmod 77 = 41$$

$$c_C = 13^3 \bmod 221 = 208$$

Attacco Low Exponent

- Esempio con m=13
  - Qualcuno riesce a intercettare i tre messaggi…
  - Risolve il sistema tramite il teorema cinese del resto

$$\begin{cases} x \equiv 7 \bmod 15 \\ x \equiv 41 \bmod 77 \\ x \equiv 208 \bmod 221 \end{cases}$$

Attacco Low Exponent

- Esempio con m=13

$$N = 15 \cdot 77 \cdot 221 = 255255$$

$$N_1 = \frac{255255}{15} = 17017 \quad Y_1 = 17017^{-1} \bmod 15 = 13$$

$$N_2 = \frac{255255}{77} 3315 \quad Y_2 = 3315^{-1} \bmod 77 = 58$$

$$N_3 = \frac{255255}{221} = 1155 \quad Y_3 = 1155^{-1} \bmod 221 = 84$$

$$x = 7 \cdot 17017 \cdot 13 + 41 \cdot 3315 \cdot 58 + 208 \cdot 1155 \cdot 84 \bmod 255255 =$$

$$1548547 + 7883070 + 20180160 \bmod 255255 =$$

$$29611777 \bmod 255255 = 2197$$

  - estrae la radice cubica di x

$$\sqrt[3]{2197} = 13$$

Attacco Low Exponent

- La scelta di e = 3, o dello stesso ordine di grandezza, rende l'RSA vulnerabile ad alcuni attacchi di tipo ciphertext-only.
- Se oltre all'esponente piccolo ci sono dipendenze lineari fra le parti del testo in chiaro, l'RSA è comunque vulnerabile (Hastad).

Attacco Chosen Ciphertext

- Daniele intercetta i messaggi cifrati di Elena (C)
- Vuole calcolare $M=C^d$
- Inoltre conosce la chiave pubblica di Elena (m ed e)

Attacco Chosen Ciphertext

- Daniele sceglie un numero casuale r tale che r<n, quindi può calcolare:

$$x = r^{e} \bmod n$$

$$y = xC \bmod n$$

$$t = r^{-1} \bmod n$$

- Se $x=r^e$ *mod n* allora $r=x^d$ *mod n*, quindi $t=x^{-d}$ *mod n*

Attacco Chosen Ciphertext

- Daniele manda y a Elena chiedendogli di firmarlo
- Elena spedisce a Daniele $u=y^d \bmod n$
- Daniele può facilmente risalire ad M

$$t \cdot u \bmod n = x^{-d} y^{d} \bmod n =$$

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

Attacco Chosen Ciphertext

- Esempio con M=53, e=13, d=37, n=77
  - Daniele intercetta il messaggio di Elena

$$C = 53^{13} \bmod 77 = 25$$

  - Sceglie r=46 e calcola:

$$x = 46^{13} \bmod 77 = 74$$

$$y = 74 \cdot 25 \bmod 77 = 2$$

$$t = 46^{-1} \bmod 77 = 72$$

Attacco Chosen Ciphertext

- Esempio con M=53, e=13, d=37, n=77
  - Daniele manda y a Elena, che risponde inviandogli

$$u = 2^{37} \bmod 77 = 51$$

  - Una semplice moltiplicazione …

$$72 \cdot 51 \bmod 77 = 53$$