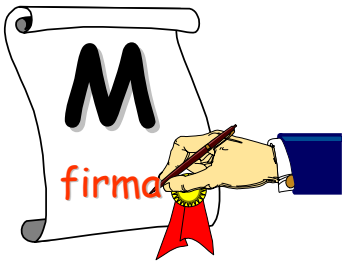
# Firme digitali

**Barbara Masucci**

Dipartimento di Informatica ed Applicazioni
Università di Salerno

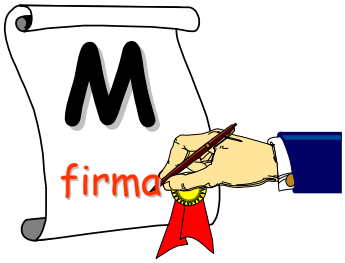
**masucci@dia.unisa.it**
**http://www.dia.unisa.it/professori/masucci**

## Firma Digitale




Equivalente alla firma convenzionale

# Firma Digitale



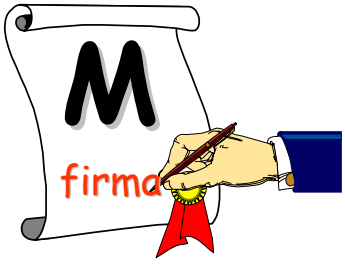

Equivalente alla firma convenzionale

Soluzione naive:

incollare firma digitalizzata



# Firma Digitale




Equivalente alla firma convenzionale


Soluzione naive:

incollare firma digitalizzata

# Requisiti per la Firma Digitale

La firma digitale deve poter essere facilmente prodotta dal legittimo firmatario

Nessun utente deve poter riprodurre la firma di altri

Chiunque può facilmente verificare una firma



# Firma digitale

chiave privata kpriv

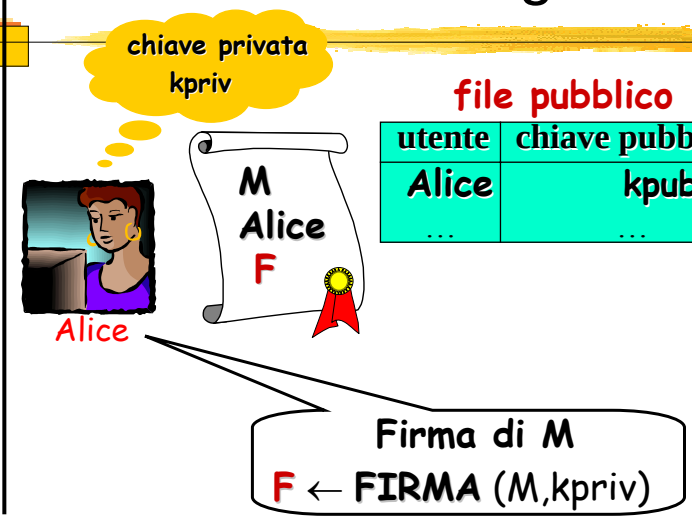
Alice


**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| … | … |

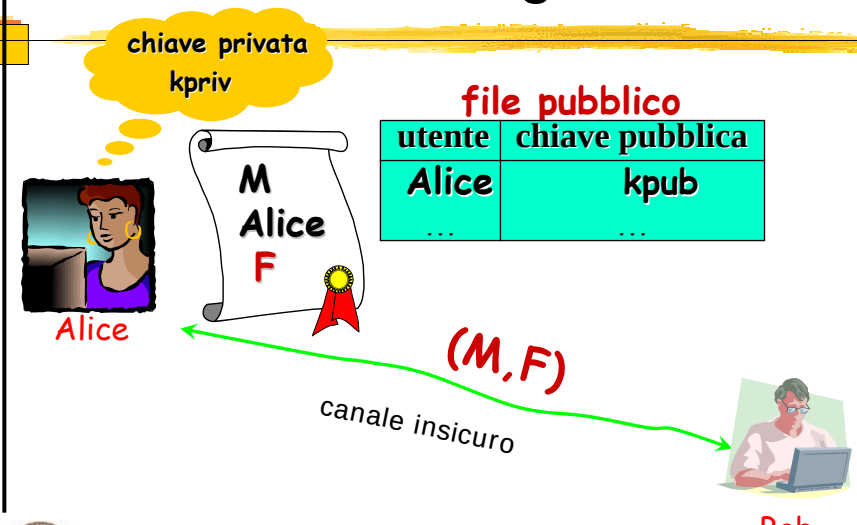
Devo firmare M

# Firma digitale






# Firma digitale



# Verifica firma digitale

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| … | … |


Devo verificare se F è una firma di Alice per M



# Verifica firma digitale

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | kpub |
| … | … |


Verifica firma di M
vera se VERIFICA (F,M,kpub) = SI
falsa altrimenti

# Sicurezza

- Cosa si intende per sicurezza di uno schema di firme digitali?
- Dobbiamo definire
  - Tipo di attacco
  - Scopo dell'attacco



# Tipo di attacco

- Key-only Attack
  - Oscar conosce solo kpub di Alice
- Known Message Attack
  - Oscar conosce una lista di messaggi e le relative firme di Alice
- Chosen Message Attack
  - Oscar sceglie dei messaggi e chiede ad Alice di firmarli

# Scopo dell'attacco

- Total break
  - Determinare kpriv di Alice per poter firmare qualsiasi messaggio

- Selective forgery
  - Dato un messaggio M, determinare la firma F tale che VERIFICA (F,M,kpub) = SI
- Existential forgery
  - Determinare una coppia (M,F) tale che VERIFICA (F,M,kpub) = SI



# Firme digitali che vedremo

- **RSA**
- **Digital Signature Standard (DSS)**


# RSA

Proposto nel 1978 da



Rivest Shamir Adleman

Sicurezza basata sulla difficoltà di fattorizzare



# Chiavi RSA

chiave privata
(n,d)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

Alice

# Chiavi RSA

chiave privata (n,d)

Alice

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| ... | ... |

n = pq
p,q primi

ed = 1 mod (p-1)(q-1)

Barbara Masucci - DIA – Università di Salerno 16

# Firma RSA

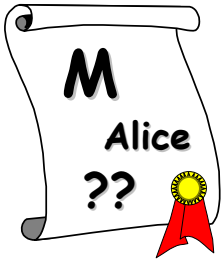
chiave privata (n,d)

Alice

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| ... | ... |

Devo firmare M




Barbara Masucci - DIA – Università di Salerno 17

# Firma RSA

chiave privata
(n,d)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

Alice

Firma di M
F ← $M^d$ mod n


Barbara Masucci - DIA – Università di Salerno

18

# Verifica Firma RSA

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |


Devo verificare se F
è una firma di Alice per M



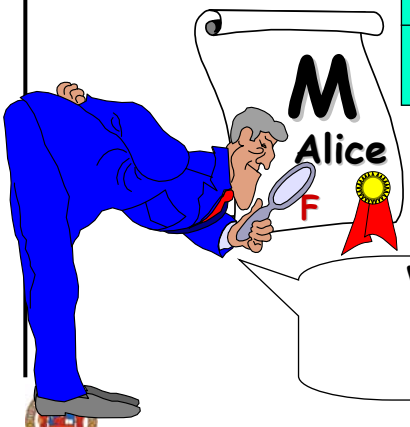
Barbara Masucci - DIA – Università di Salerno

19

# Verifica Firma RSA

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |




**Verifica firma di M**
vera se $M = F^e \bmod n$
falsa altrimenti



# "Piccolo" esempio: Chiavi RSA

**chiave privata**
**(n=3337, d=1019)**

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n = 3337, e = 79) |
| … | … |

$3337 = 47 \cdot 71$
$p = 47, q = 71$

$ed = 79 \cdot 1019 = 1 \bmod 3220$
$(p-1)(q-1) = 46 \cdot 70 = 3220$

Alice

# "Piccolo" esempio: Chiavi RSA

chiave privata (n=3337, d=1019)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n = 3337, e = 79) |
| … | … |

Alice

Devo firmare M=1570

1570
Alice

Barbara Masucci - DIA – Università di Salerno

22

# "Piccolo" esempio: generazione firma RSA

chiave privata (n=3337, d=1019)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n = 3337, e = 79) |
| … | … |

Alice

Firma di 1570
= $1570^{1019}$ mod 3337
= 668

1570
Alice
668

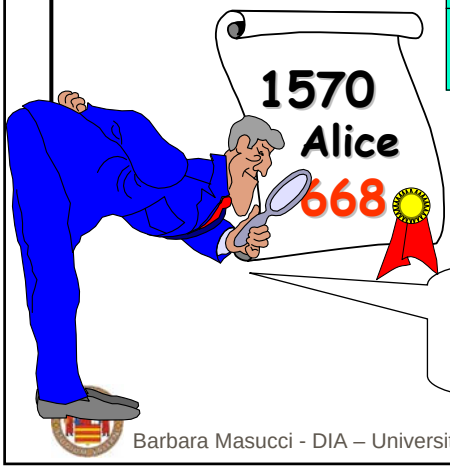
Barbara Masucci - DIA – Università di Salerno

23

# "Piccolo" esempio: Verifica firma RSA

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (n = 3337, e = 79) |
| … | … |




Verifica firma di 1570
$1570 = 668^{79} \mod 3337$



# Correttezza verifica firma RSA

$C^d \mod n = (M^e)^d \mod n$

$= M^{ed} \mod n$ — $ed = 1 \mod (p-1)(q-1)$

$= M^{1+r(p-1)(q-1)} \mod n$

$= M \cdot (M^r)^{(p-1)(q-1)}$ — **Teorema di Eulero** $a \in Z_n^* \Rightarrow a^{(p-1)(q-1)} = 1 \mod n$

$= M \mod n$

$= M$ — poichè $0 \le M < n$

## Sicurezza firma RSA

Voglio falsificare la firma di M da parte di A

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

Devo calcolare $M^d \bmod n$

Equivalente a "rompere" il crittosistema RSA


**Selective forgery**
**Key only attack**



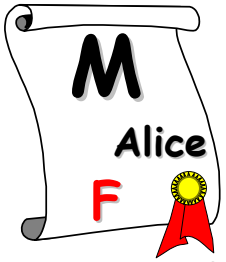
## Sicurezza firma RSA

Voglio generare messaggi e firme da parte di A

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

1. Scelgo F a caso
2. $M \leftarrow F^e \bmod n$




**Existential forgery**
**Key only attack**

## Sicurezza firma RSA

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| ... | ... |

Voglio generare messaggi e firme da parte di A

Conosco le coppie $(M_1,F_1)$ e $(M_2,F_2)$

**Proprietà di omomorfismo**

$F_1 = M_1^d \bmod n \quad F_2 = M_2^d \bmod n$

$(F_1F_2)^e \bmod n = F_1^e F_2^e \bmod n = M_1M_2 \bmod n$

$F_1F_2 \bmod n$ è una firma valida per $M_1M_2 \bmod n$

**Existential forgery**

**Known message attack**

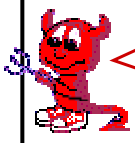


## Sicurezza firma RSA

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| ... | ... |

Voglio falsificare la firma di M da parte di A



1. Scelgo $M_1$ e $M_2$ tali che $M=M_1M_2 \bmod n$
2. Chiedo ad Alice di firmare $M_1$ e $M_2$ ottenendo $F_1$ e $F_2$
3. $F_1F_2 \bmod n$ è una firma valida per M

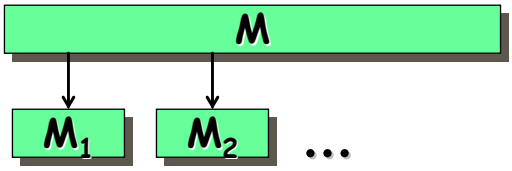
**Selective forgery**

**Chosen message attack**

# Firma digitale di messaggi grandi

Se M>n, come si firma?

Prima soluzione

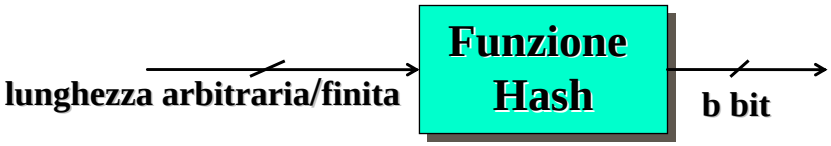



Firma(M) ← (Firma($M_1$), Firma($M_2$), ...)

**Problemi**
- Efficienza
- Permutazione/composizione delle firme → nuova firma



# Funzioni Hash




Il valore hash h(M) è una rappresentazione non ambigua e non falsificabile del messaggio M

Proprietà:
- comprime
- facile da computare
- **Sicurezza forte**: computazionalmente difficile trovare 2 diversi messaggi con lo stesso valore hash
- **One-way**: dato y è computazionalmente difficile trovare M tale che y = h(M)

# Firma digitale con hash

messaggi piccoli

M → h(M)

messaggi grandi

M → h(M)

Firma(M) ← (Firma(h(M))

**Vantaggi**
- Efficienza
- Integrità
- Sicurezza



# Firma RSA con hash

chiave privata (n,d)

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

Alice

**Firma di M**

F ← $[h(M)]^d \bmod n$

M Alice F

# Verifica Firma RSA con hash

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

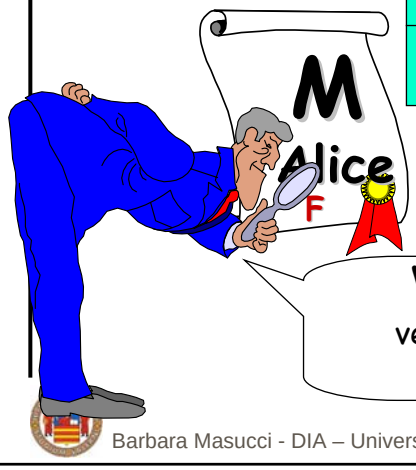

Devo verificare se F è una firma di Alice per M



# Verifica Firma RSA

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |




**Verifica firma di M**

vera se $h(M) = F^{e} \bmod n$

falsa altrimenti

# Sicurezza firma RSA con hash

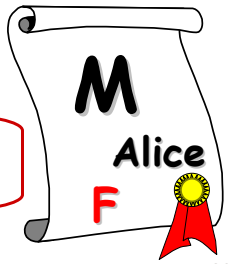
Voglio generare messaggi e firme da parte di A

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (n,e) |
| … | … |

1. Scelgo F a caso
2. $z \leftarrow F^e \bmod n$
3. $M \leftarrow h^{-1}(z)$

Come faccio ad invertire h?
$M \leftarrow h^{-1}(z)$




**Existential forgery**

**Key only attack**



# Digital Signature Standard (DSS)

- Proposto nell'agosto del 1991 dal NIST
  - Digital Signature Algorithm (DSA)
  - Digital Signature Standard (DSS)
- Standard rivisto nel 1993, in risposta alle critiche
- Modifica ingegnosa dello schema di firme El Gamal
- Utilizza la funzione hash SHA (message digest di 160 bit)
- Firme DSS sempre di 320 bit (buone per smart card)
- Sicurezza basata sull'intrattabilità del problema del logaritmo discreto

# Chiavi DSA

chiave privata
(p,q,α,s)

Alice

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (p,q,α,β) |
| … | … |



# Chiavi DSA

chiave privata
(p,q,α,s)

Alice

**file pubblico**

| utente | chiave pubblica |
|---|---|
| Alice | (p,q,α,β) |
| … | … |

p primo di L bit
512 ≤ L ≤ 1024, L multiplo di 64

β=α^s mod p

α in $Z_p^*$ di ordine q

q primo di 160 bit, q|(p-1)

s numero casuale, s<q

α^q = 1 mod p

## Ordine di un elemento

- Per la generazione di $\alpha$ dobbiamo introdurre il concetto di ordine di un elemento
- Ordine di $\alpha \in Z_n^*$ = il più piccolo intero positivo r tale che $\alpha^r = 1 \bmod n$
- Teorema di Lagrange:
  - Per ogni $\alpha \in Z_n^*$, ord($\alpha$) divide $\phi(n)$
    - Se p è primo, ord($\alpha$) divide p-1



## Ordine di un elemento

Esempio: n=15, $\phi(n) = (3-1)(5-1)=8$

$Z_{15}^* = \{ 1, 2, 4, 7, 8, 11, 13, 14 \}$

- ord(1) = 1, ord(2) = 4,
- ord(4) = 2, ord(7) = 4,
- ord(8) = 4, ord(11) = 2,
- ord(13) = 4, ord(14) = 2.

# Ordine di un elemento

Sia $\alpha \in Z_n^*$ e sia $q = ord(\alpha)$

- Se $\alpha^t = 1 \bmod p$, allora $q$ divide $t$
- $\alpha^{s \bmod q} \bmod p = \alpha^s \bmod p$



# Chiavi DSA ("piccolo" esempio)


file pubblico

| utente | chiave pubblica |
|---|---|
| A | (7879,101,170,4567) |
| ... | ... |

p = 7879 primo

4567 = $170^{75} \bmod 7879$

$\alpha = 170 \in Z_{7879}^*$ di ordine 101

q = 101 primo, p = 78q+1

$170^{101} = 1 \bmod 7879$

s = 75 numero casuale

# Firma DSA

chiave privata
(p,q,α,s)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (p,q,α,β) |
| … | … |

Alice

Devo firmare M




# Firma DSA

chiave privata
(p,q,α,s)

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (p,q,α,β) |
| … | … |

Alice

**Firma di M**

r ← numero casuale in [1,q-1]

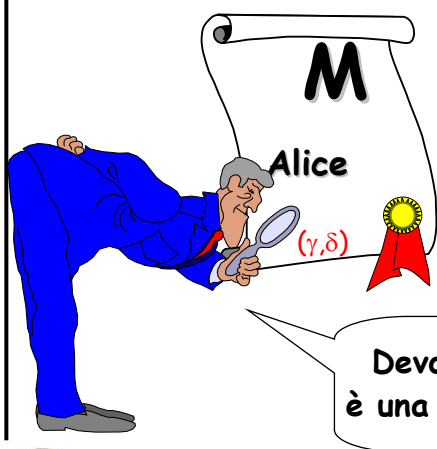
$\gamma \leftarrow (\alpha^r \bmod p) \bmod q$

$\delta \leftarrow (SHA(M)+s\gamma)r^{-1} \bmod q$

$firma_{(p,q,\alpha,s)}(M,r) = (\gamma,\delta)$


$r^{-1} \bmod q$ esiste perché
$r < q$ e q primo → gcd(q,r)=1

# Verifica firma DSA



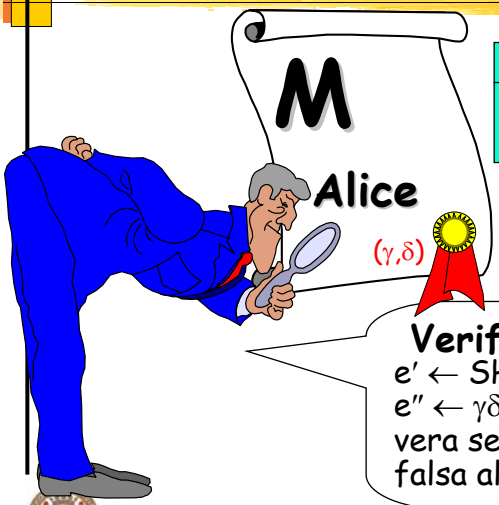

file pubblico

| utente | chiave pubblica |
|---|---|
| Alice | (p,q,α,β) |
| … | … |

Devo verificare se (γ,δ) è una firma di Alice per M



# Verifica firma DSA




file pubblico

| utente | chiave pubblica |
|---|---|
| A | (p,q,α,β) |
| … | … |

**Verifica firma di M**

$e' \leftarrow SHA(M)\delta^{-1} \bmod q$

$e'' \leftarrow \gamma\delta^{-1} \bmod q$

vera se $\gamma = (\alpha^{e'}\beta^{e''} \bmod p) \bmod q$

falsa altrimenti

# Efficienza firma DSA

**Firma_DSA**$(M,p,q,\alpha,s)$
$r \leftarrow$ numero casuale in $[1,q-1]$
$\gamma \leftarrow (\alpha^r \bmod p) \bmod q$
$\delta \leftarrow (SHA(M)+s\gamma)r^{-1} \bmod q$
**output** $firma_{(p,q,\alpha,s)}(M,r) = (\gamma,\delta)$

- Lunghezza firma = 320 bit
- Computazioni off-line: $r$, $s\gamma$, $r^{-1} \bmod q$
- Computazioni on-line: $SHA(M)$, $+$, $\cdot$



# Verifica firma DSA

**Verifica_firma_DSA**$(M,\gamma,\delta,p,q,\alpha,\beta)$
$e' \leftarrow SHA(M)\delta^{-1} \bmod q$
$e'' \leftarrow \gamma\delta^{-1} \bmod q$

$$ver_{(p,q,\alpha,\beta)}(M,\gamma,\delta) = \begin{cases} \text{vera se } \gamma = (\alpha^{e'}\beta^{e''} \bmod p) \bmod q \\ \text{falsa altrimenti} \end{cases}$$

**Output** $ver_{(p,q,\alpha,\beta)}(M,\gamma,\delta)$

## Correttezza verifica firma DSA

$e' = SHA(M)\delta^{-1} \bmod q$

$e'' = \gamma\delta^{-1} \bmod q$

$\beta = \alpha^{s} \bmod p$

$$(\alpha^{e'}\beta^{e''} \bmod p) \bmod q$$

$$= (\alpha^{SHA(M)\delta^{-1} \bmod q}\ \alpha^{s\gamma\delta^{-1} \bmod q} \bmod p) \bmod q$$

$\alpha$ è di ordine q

$$= (\alpha^{SHA(M)\delta^{-1}+s\gamma\delta^{-1}} \bmod p) \bmod q$$

$\delta^{-1}(SHA(M)+s\gamma) = r \bmod q$

$$= (\alpha^{r} \bmod p) \bmod q$$

$$= \gamma$$



## Scelta dei parametri

- Come scegliere p, q, $\alpha$ ?

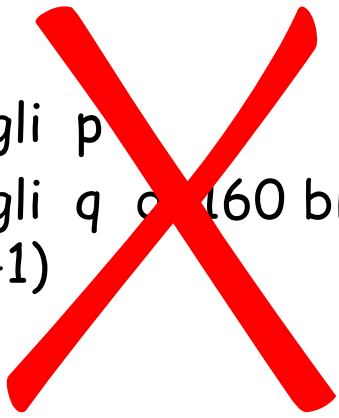
## Generazione di p e q

- Scegli p
- Scegli q di 160 bit tale che q|(p-1)



## Generazione di p e q

- Scegli p
- Scegli q di 160 bit tale che q|(p-1)




# Generazione di p e q

- Scegli un primo q di 160 bit
- Scegli un primo p di 512/1024 bit tale che q|(p-1)
  - Scegli X di 512 bit (oppure … 1024 bit)
  - p ← X - ((X mod 2q)-1)
  - se p è primo e $p \geq 2^{511}$ esci altrimenti riprova

2q|(p-1)

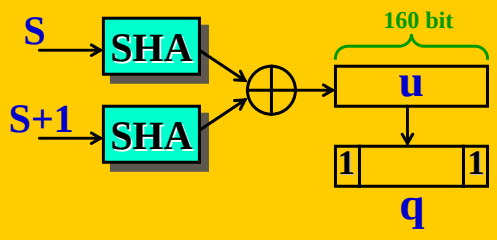


# Generazione di p e q

```
Seleziona_pq(L)
(1) Computa interi n e b tali che L-1=160n+b
(2) repeat
(3)     repeat
(4)         S ← sequenza casuale di almeno 160 bit
(5)         g ← |S|
(6)         U ← SHA(S)⊕ SHA((S+1) mod 2^g)
(7)         Forma q da U ponendo il MSB ed il LSB ad 1
(8)     until q primo
(9)     C ← 0
(10)    N ← 2
(11)    repeat
(12)        for k=0 to n do V_k ← SHA(S+N+k) mod 2^g
(13)        W ← V_0+V_1·2^160+...+V_n-1·2^160(n-1)+(V_n mod 2^b)·2^160n
(14)        X ← W+2^L-1
(15)        p ← X-((X mod 2q)-1)
(16)    until (p primo) or (p<2^L-1)
(17)    if p<2^L-1
(18)        then C ← C+1
(19)           N ← N+n+1
(20)           if C<4096 then goto step (12)
(21)                     else Help ← falso
(22)        else Help ← vero
(23) until Help
(24) return p,q,S,C
```

# Generazione di q

Scegli a caso S di ≥160 bit



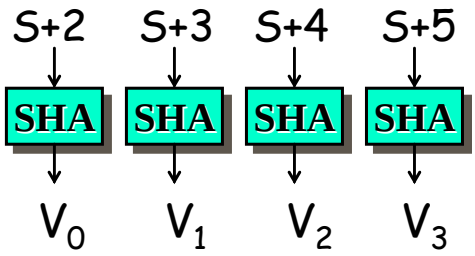

Ripeti con un nuovo S finchè q è primo



# Generazione di p e q

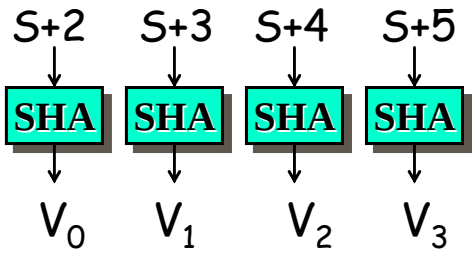
```
Seleziona_pq(L)
(1) Computa interi n e b tali che L-1=160n+b
(2) repeat
(3)     repeat
(4)         S ← sequenza casuale di almeno 160 bit
(5)         g ← |S|
(6)         U ← SHA(S)⊕ SHA((S+1) mod 2^g)
(7)         Forma q da U ponendo il MSB ed il LSB ad 1
(8)     until q primo
(9)     C ← 0
(10)    N ← 2
(11)    repeat
(12)        for k=0 to n do V_k ← SHA(S+N+k) mod 2^g
(13)        W ← V_0+V_1·2^160+...+V_n-1·2^160(n-1)+(V_n mod 2^b)·2^160n
(14)        X ← W+2^L-1
(15)        p ← X-((X mod 2q)-1)
(16)    until (p primo) or (p<2^L-1)
(17)    if p<2^L-1
(18)        then C ← C+1
(19)           N ← N+n+1
(20)           if C<4096 then goto step (12)
(21)                     else Help ← falso
(22)        else Help ← vero
(23) until Help
(24) return p,q,S,C
```

## Generazione di p (512 bit)






## Generazione di p (512 bit)




$511 = 3 \cdot 160 + 31$

$|X|=512$

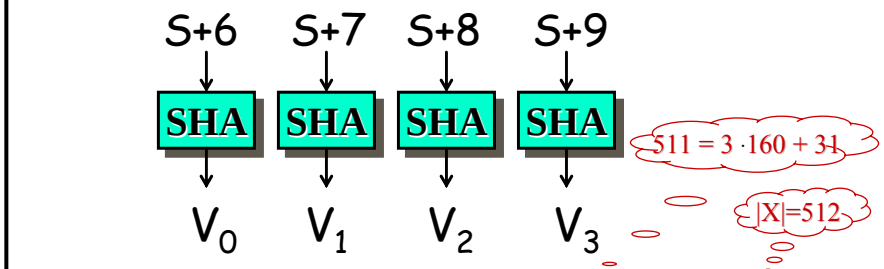
$X \leftarrow V_0 + V_1 \cdot 2^{160} + V_2 \cdot 2^{2 \cdot 160} + (V_3 \bmod 2^{31}) \cdot 2^{3 \cdot 160} + 2^{511}$

$p \leftarrow X - ((X \bmod 2q)-1)$

$2q|(p-1)$

se p è primo e $p \geq 2^{511}$ esci altrimenti …

## Generazione di p (512 bit)




$X \leftarrow V_0 + V_1 \cdot 2^{160} + V_2 \cdot 2^{2 \cdot 160} + (V_3 \bmod 2^{31}) \cdot 2^{3 \cdot 160} + 2^{511}$

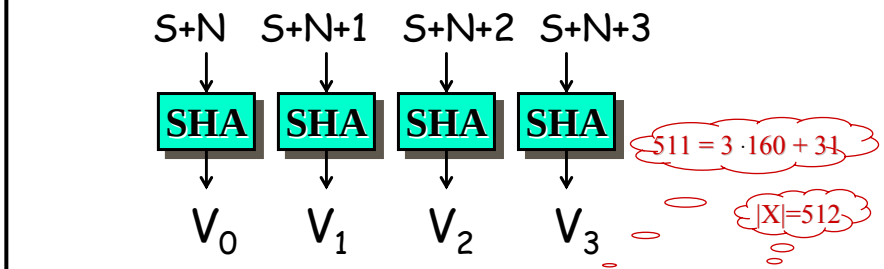
$p \leftarrow X - ((X \bmod 2q) - 1)$ — 2q|(p-1)

se p è primo e $p \geq 2^{511}$ esci altrimenti …



## Generazione di p (512 bit)

**N←2,6,10,…, (per ≤4096 volte)**




$X \leftarrow V_0 + V_1 \cdot 2^{160} + V_2 \cdot 2^{2 \cdot 160} + (V_3 \bmod 2^{31}) \cdot 2^{3 \cdot 160} + 2^{511}$

$p \leftarrow X - ((X \bmod 2q) - 1)$ — 2q|(p-1)

se p è primo e $p \geq 2^{511}$ esci altrimenti …

## Scelta di un elemento di ordine q

- p,q primi tali che q|(p-1)

**Scegli_ordineq** (p,q)
1. g ← elemento scelto a caso in $Z_p^*$
2. $\alpha \leftarrow g^{(p-1)/q} \bmod p$
3. **if** $\alpha \neq 1$ **then return** $\alpha$ **else** go to 1.



## Correttezza di Scegli_ordineq

**Scegli_ordineq** (p,q)
1. g ← elemento scelto a caso in $Z_p^*$
2. $\alpha \leftarrow g^{(p-1)/q} \bmod p$
3. if $\alpha \neq 1$ then return $\alpha$ else go to 1.

- $\alpha^q \equiv (g^{(p-1)/q})^q \equiv g^{p-1} \equiv 1 \bmod p$
- Quindi ord($\alpha$) divide q
  - q primo → ord($\alpha$)=1 oppure ord($\alpha$)=q
    - ord($\alpha$)=1 sse $\alpha$ =1
    - $\alpha \neq 1$ → ord($\alpha$)=q

## Probabilità successo

- Se g è un generatore allora $g^{(p-1)/q} \neq 1 \bmod p$
- Probabilità successo $\geq$ Probabilità che g è generatore
  $> 1/(6\ln\ln(p-1))$
- Numero medio di iterazioni $< 6\ln\ln(p-1)$

| | |
|---|---|
| 512 bit | $6 \cdot \ln\ln(2^{512}) \approx 35{,}23$ |
| 1024 bit | $6 \cdot \ln\ln(2^{1024}) \approx 39{,}38$ |
| 2048 bit | $6 \cdot \ln\ln(2^{2048}) \approx 43{,}54$ |



## Sicurezza firma DSA

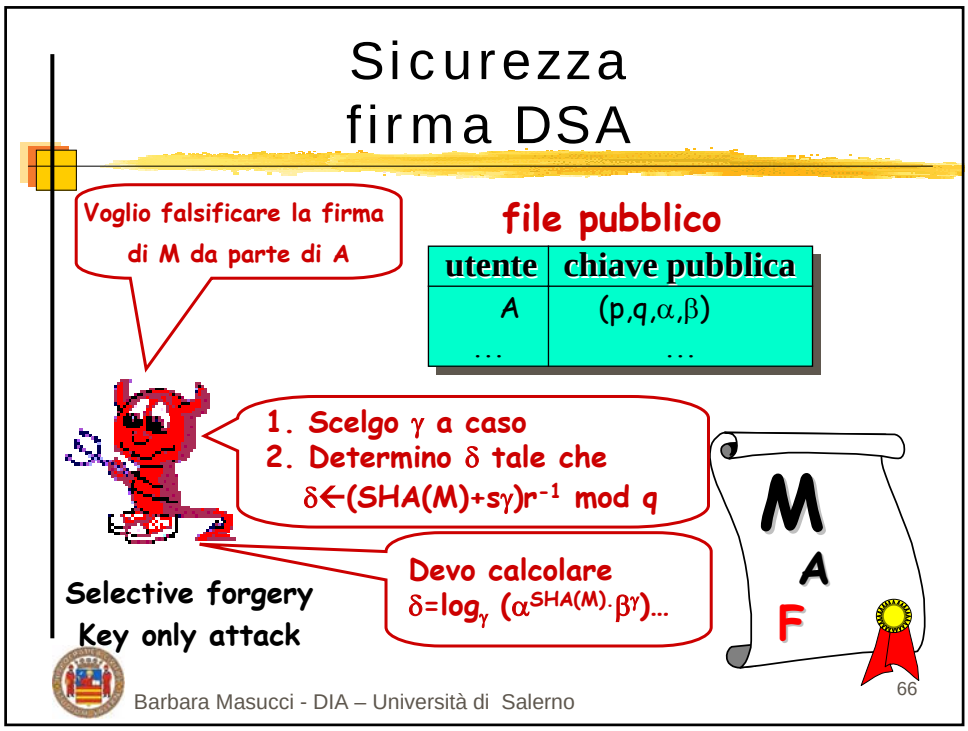
Voglio falsificare la firma di M da parte di A

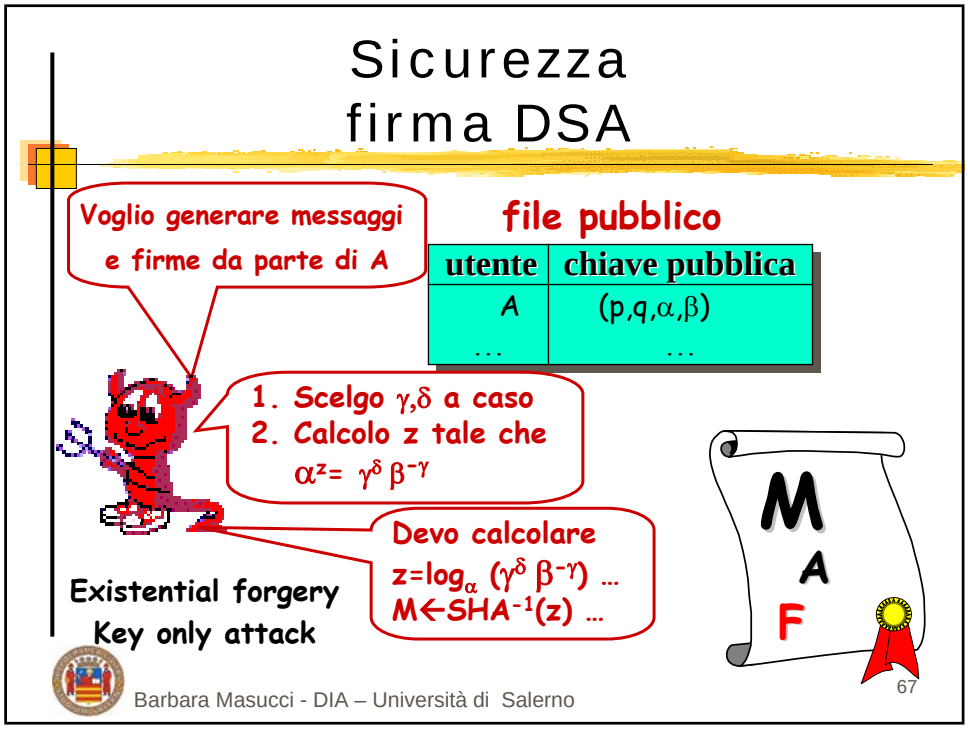
**file pubblico**

| utente | chiave pubblica |
|---|---|
| A | $(p,q,\alpha,\beta)$ |
| … | … |

Devo calcolare $s=\log_{\alpha}\beta \bmod p$…


**Total break**
**Key only attack**


Sicurezza
firma DSA
Voglio falsificare la firma di M da parte di A
file pubblico
utente
chiave pubblica
A
(p,q,α,β)
1. Scelgo γ a caso
2. Determino δ tale che
δ←(SHA(M)+sγ)r^-1 mod q
Devo calcolare
δ=log_γ (α^SHA(M)·β^γ)...
M
A
F
Selective forgery
Key only attack
Barbara Masucci - DIA – Università di Salerno
66


Sicurezza
firma DSA
Voglio generare messaggi e firme da parte di A
file pubblico
utente
chiave pubblica
A
(p,q,α,β)
1. Scelgo γ,δ a caso
2. Calcolo z tale che
α^z= γ^δ β^-γ
Devo calcolare
z=log_α (γ^δ β^-γ) ...
M←SHA^-1(z) ...
M
A
F
Existential forgery
Key only attack
Barbara Masucci - DIA – Università di Salerno
67

# Chiavi globali ed individuali

chiave privata (p,q,α,s)

**file pubblico**

| utente | chiave pubblica |
|---|---|
| A | (p,q,α,β) |
| ... | ... |

Alice

- Sicurezza basata sul valore privato s
- I valori p,q,α possono essere gli stessi per un gruppo di utenti
- Un'autorità sceglie p,q,α
- Il singolo utente sceglie solo s e calcola β



# Confronto tempi firme RSA e DSA

| | DSA | RSA | DSA con p,q,α comuni |
|---|---|---|---|
| precomputazioni | 14 sec | | 4 sec |
| firma | 0.3 sec | 15 sec | 0.3 sec |
| verifica | 16 sec | 1.5 sec | 10 sec |
| | 1-5 sec Off Cards | | 1-3 sec Off Cards |

- Implementazioni su smart card [1993]
- Computazioni Off Cards su 80386 a 33MHz

# Prestazioni algoritmi

Celeron 850MHz, Windows 2000, Crypto++

millisecondi/operazione

| | bit chiave | firma | Firma con precomputazione | verifica |
|---|---|---|---|---|
| RSA | 512 | 1,92 | | 0,13 |
| DSA | 512 | 1,77 | 1,19 | 2,02 |
| RSA | 1024 | 10,29 | | 0,30 |
| DSA | 1024 | 5,50 | 2,27 | 6,38 |



# Prestazioni

Pentium II 400
OpenSSL

| | bit chiave | firme/s | verifiche/s |
|---|---|---|---|
| RSA | 512 | 342 | 3287 |
| DSA | 512 | 331 | 273 |
| RSA | 1024 | 62 | 1078 |
| DSA | 1024 | 112 | 94 |
| RSA | 2048 | 10 | 320 |
| DSA | 2048 | 34 | 27 |

# Bibliografia

- Cryptography and Network Security by W. Stallings (2003)
  - cap. 12 (DSS)
- Tesina di Sicurezza su reti
  - Firme digitali
- Stinson I ed