Sicurezza Informatica
Firma Digitale

RSA

RSA-Embedding

El Gamal

Limiti cifratura simmetrica

C=Enc("Alice è in debito con Bob di 100€", sk)

- Poiché la chiave simmetrica è condivisa da mittente e destinatario non si può essere sicuri dell'autenticità del messaggio cifrato
  - Non si riesce a determinare chi tra mittente e destinatario abbia generato il messaggio
  - Chiunque in possesso della chiave simmetrica sk può aver generato il messaggio
  - Un giudice non potrebbe se il contratto C sia valido oppure no

Requisiti firma digitale

Analogie con il mondo analogico

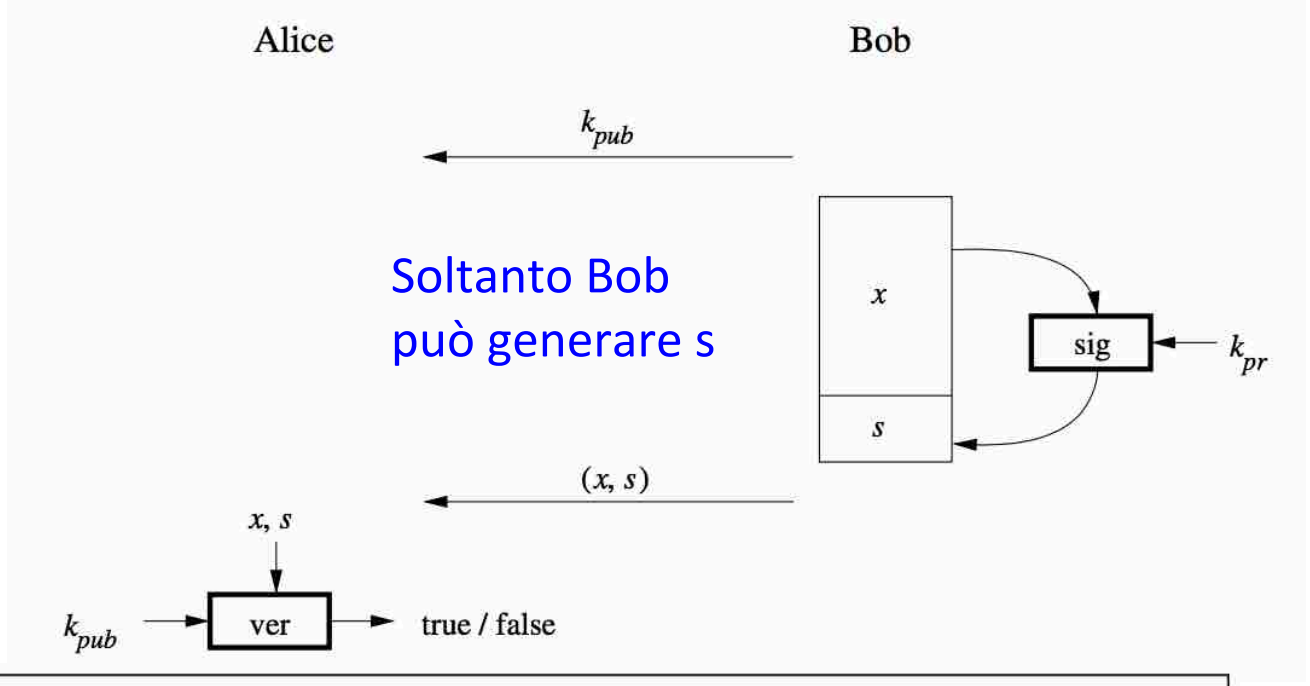
- La firma digitale deve poter esser facilmente prodotta dal legittimo firmatario
- La firma deve cambiare per ogni documento
- Nessun utente deve poter riprodurre la firma di altri utenti
- Chiunque può facilmente verificare una firma digitale
- Deve essere possibile provare che un firmatario ha effettivamente firmato un documento
  - La prova potrebbe avere valore legale
    - Dipende dalla normativa vigente

Firma Digitale

- È una stringa di bit che dipenda da un segreto noto solo al firmatario e dal messaggio che si firma

- Una firma digitale deve essere verificabile, nel caso di disputa, da una terza parte senza richiedere l'accesso al segreto conservato dal firmatario

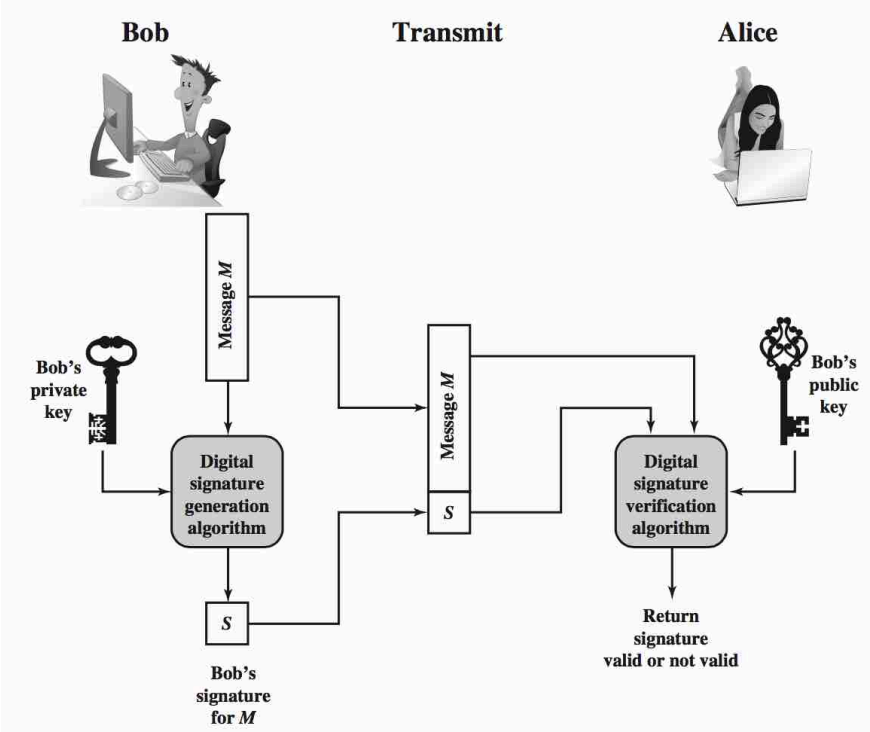
Principi della firma digitale



Basic Digital Signature Protocol

Alice		Bob
		generate $k_{pr,B}$, $k_{pub,B}$
	$k_{pub,B}$ ←	publish public key
		sign message: $s = \text{sig}_{k_{pr}}(x)$
	(x,s) ←	send message + signature
verify signature: $\text{ver}_{k_{pr,B}}(x,s) = \text{true/false}$		

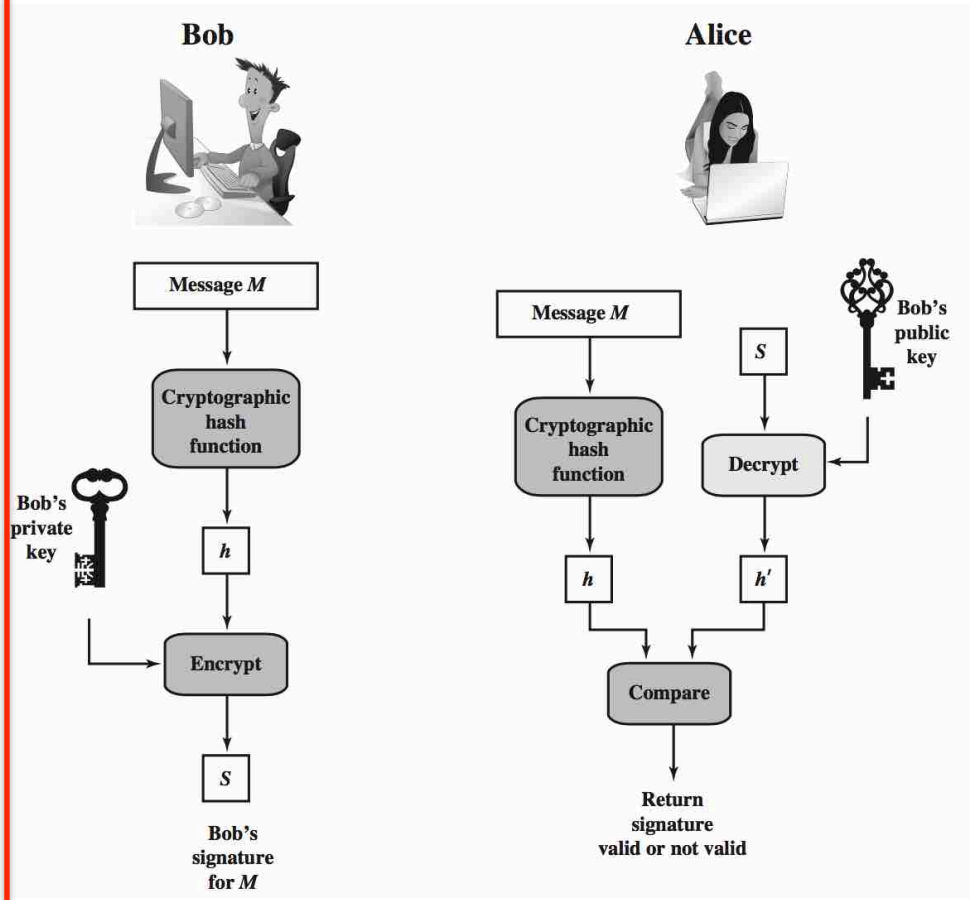
Modello generico



Possiamo aggiungere confidenzialità cifrando il messaggio M

Soluzione possibile?

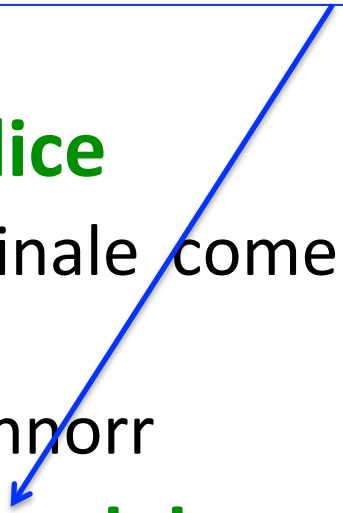
Usiamo schema di cifratura a chiave pubblica



OK se Enc e Dec sono *commutative*

Tipi firme

Embedding del messaggio in un messaggio EM che sarà effettivamente firmato
e.g., EM = M || M oppure EM = M || H(M)



- **Firma digitale con appendice**
  - Richiede il messaggio originale come input della procedura di verifica
  - ElGamal, RSA, DSA, DSS, Schnorr
- **Firma digitale con recupero del messaggio**
  - Non richiede il messaggio originale come input della procedura di verifica
  - Il messaggio è recuperato dalla firma stessa
  - RSA (modificato), Rabin, Nyberg-Rueppel

Servizi di sicurezza desiderabili

- **Confidentiality** (Confidenzialità)
  - L'informazione è tenuta segreta a tutti tranne che per le entità autorizzate
- **Integrity** (Integrità)
  - Assicura che un messaggio in transito non è stato modificato
- **Message Authentication** (Autenticazione del messaggio)
  - Assicura che il mittente di un messaggio è autentico Un altro modo di chiamare il servizio è *data origin authentication*
- **Non-repudiation** (Non ripudio)
  - Assicura che il mittente di un messaggio non può negare la creazione del messaggio stesso

Ottenibili tramite l'utilizzo della firma digitale

Tipi di attacchi – 1

- **Key-only attack**
  - L'attaccante conosce solo la chiave pubblica di Alice
- **Known message attack**
  - L'attaccante conosce un insieme di messaggi e le rispettive firme
- **Generic chosen message attack**
  - L'attaccante sceglie una lista di messaggi non conoscendo la chiave pubblica di Alice. L'attaccante ottiene le firme dei messaggi che ha scelto

L'attacco è generico perché non dipende dalla chiave pubblica di Alice

Tipi di attacchi – 2

- **Directed chosen message attack**
  - Simile all'attacco generico, ma la lista dei messaggi da far firmare è scelta dopo che l'attaccante conosce la chiave pubblica di Alice ma prima di conoscere una qualsiasi firma
- **Adaptive chosen message attack**
  - L'attaccante può utilizzare Alice come un oracolo (l'attaccante può richiedere ad Alice una firma di un messaggio che dipende dalle coppie di messaggio-firma che ha ottenuto in precedenza

L'avversario ha accesso ad un *oracolo di firma*

Tipi rottura schema - 1

- **Total Break**
  - L'attaccante determina la chiave privata di Alice

- **Universal forgery**
  - L'attaccante trova un algoritmo di firma efficiente che fornisce un modo equivalente di costruire firme di un qualsiasi messaggio (non richiede la conoscenza della chiave privata di Alice)

Tipi rottura schema - 2

- **Selective forgery**
  - L'attaccante riesce a falsificare una firma per un messaggio particolare scelto dall'attaccante stesso

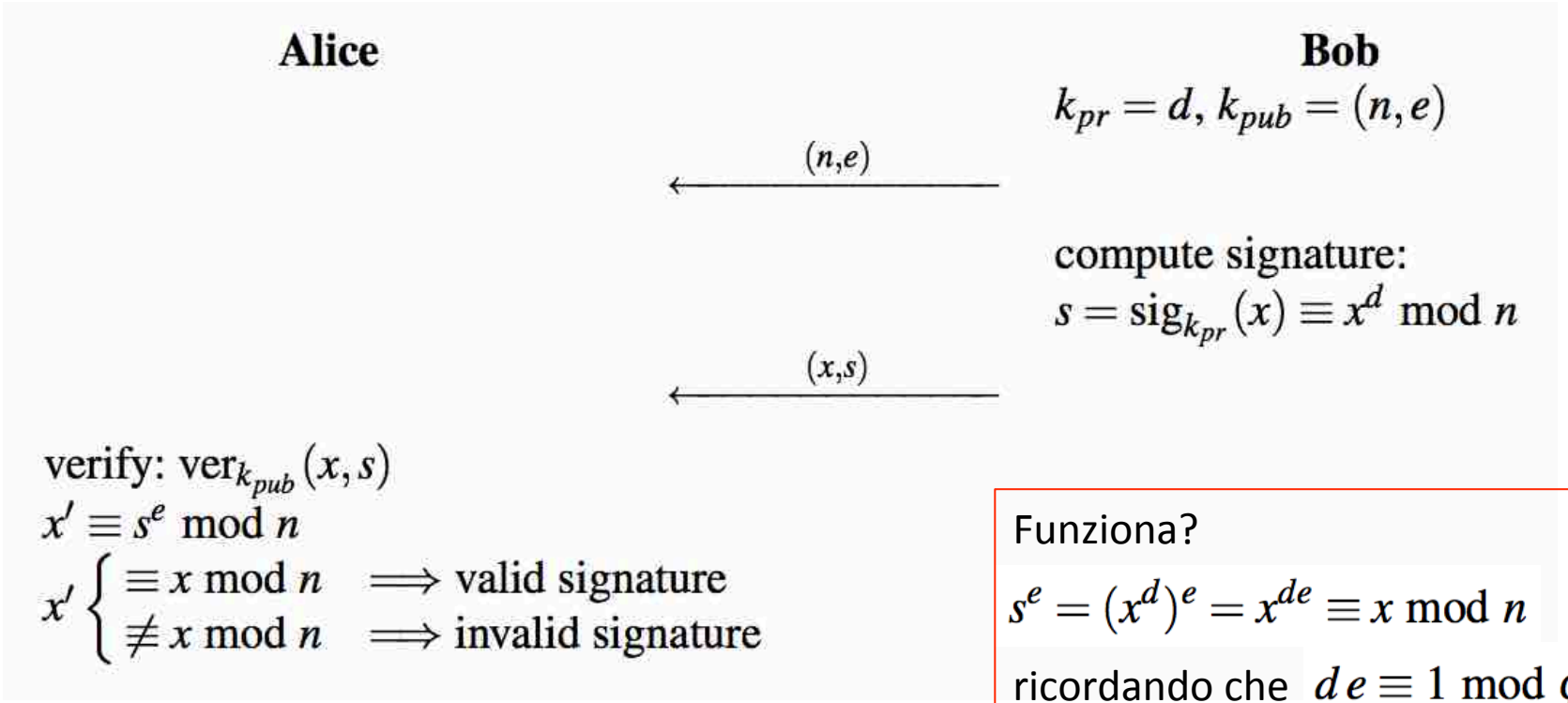
- **Existential forgery**
  - L'attaccante riesce a falsificare una firma per almeno un messaggio. L'attaccante non ha controllo su quale messaggio riesce a firmare
  - È l'attacco che nuoce meno ad Alice

Firma Digitale RSA (*schoolbook*)

RSA Keys

- Bob's private key: $k_{pr} = (d)$
- Bob's public key: $k_{pub} = (n, e)$

Assumiamo che x∈ [1,…,n-1]



Funziona?

$s^e = (x^d)^e = x^{de} \equiv x \bmod n$

ricordando che $d\,e \equiv 1 \bmod \phi(n)$

Cosa ci dice x=x'?

- Se x = x', Alice deduce che l'autore del messaggio era in possesso della chiave privata di Bob
  - Message Authentication
- Il messaggio non è stato modificato durante il transito sulla rete
  - Message Integrity

Sicurezza firma RSA

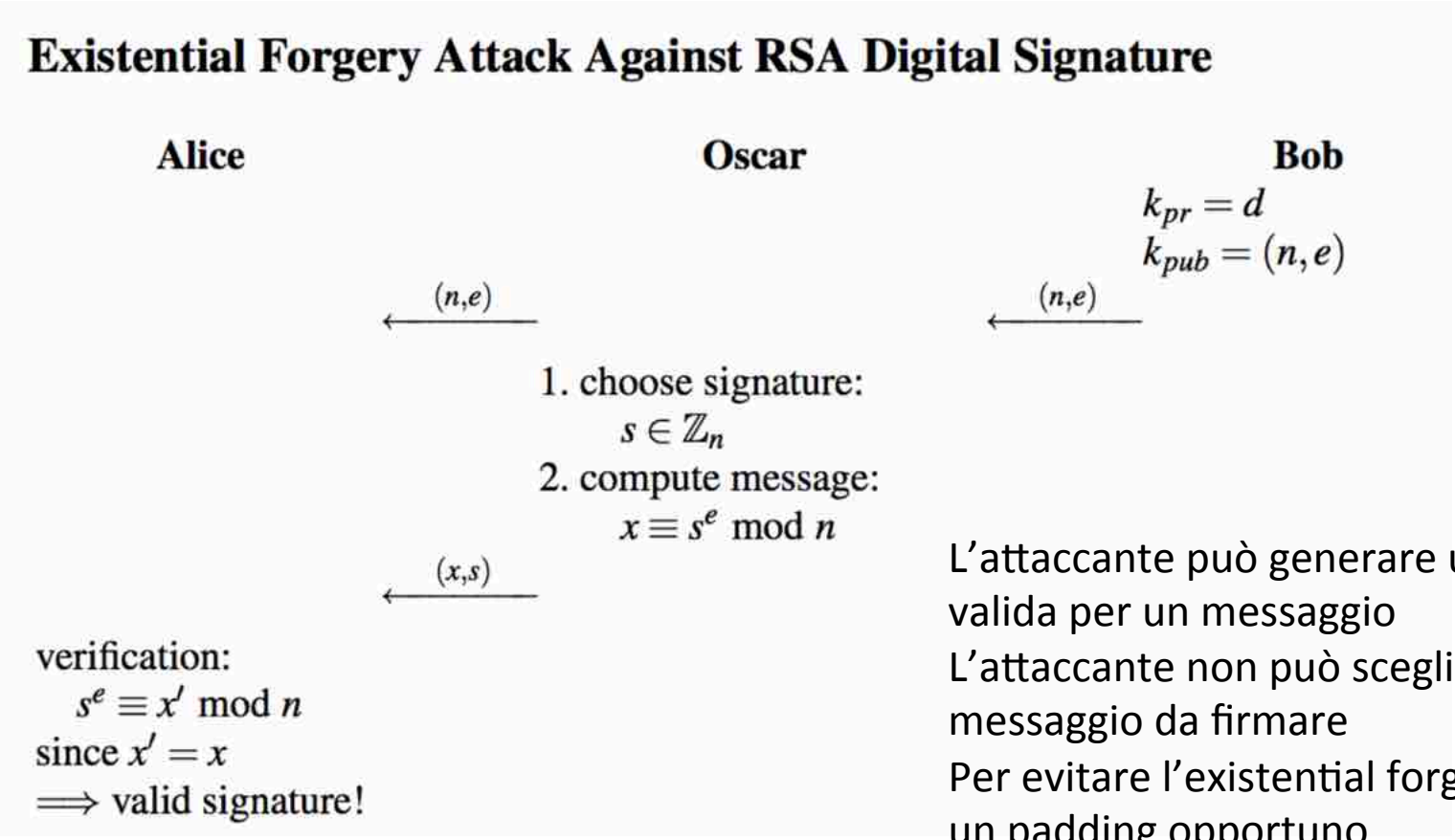
- Stessi argomenti della cifratura RSA
  - Fattorizzazione di n implica recupero chiave d
- Per avere una sicurezza di 80 bit è necessario che n sia di almeno 1024 bit

- La dimensione della firma s sarà di almeno 1024 bit

Aspetti computazionali

- Firma e verifica consistono in un elevamento a potenza modulo n
- Tipicamente n varia da 1024 a 3072 bit
- In pratica si usa e = $2^{16}+1$, ciò rende la procedura di verifica molto efficiente
  - A scapito della firma
  - Non è un problema in quanto si firma una sola volta un messaggio, ma si verifica la firma molte volte

Existential Forgery per RSA



L'attaccante può generare una firma valida per un messaggio
L'attaccante non può scegliere il messaggio da firmare
Per evitare l'existential forgery si utilizza un padding opportuno

Key-only Attack

Existential Forgery
Know Message Attack

- L'attaccante conosce le coppie (M_1, F_1) e (M_2, F_2)
  - Ricordiamo che $F_1 = M_1^d \bmod n$ e $F_2 = M_2^d \bmod n$

- L'avversario calcola $(M_1 \cdot M_2, F_1 \cdot F_2)$ che è uguale a $(M_1 \cdot M_2, M_1^d \cdot M_2^d) = (M_1 \cdot M_2, (M_1 \cdot M_2)^d)$

- L'avversario ottiene una firma $F_1 \cdot F_2 \bmod n$ valida per il messaggio $M_1 \cdot M_2$

Selective Forgery
Chosen Message Attack

- L'avversario vuole falsificare la firma del messaggio M.. Sceglie M_1 e M_2 tali che $M=M_1 \cdot M_2$

- L'avversario chiede all'oracolo di firma di firmare i messaggi M_1 e M_2 ottenendo F_1 e F_2

- $F_1 \cdot F_2$ mod n è una firma valida per il messaggio $M_1 \cdot M_2$

Firma di messaggi grandi

- Se M > n come possiamo firmare il messaggio?
- Suddividiamo il messaggio M in M_1, M_2, M_3, …, con M_i<n, e firmiamo singolarmente ogni M_i
- Firma(M) = [Firma(M_1), Firma(M_2), Firma(M_3), …]

- Problemi
  - Efficienza
  - Permutazione/Composizione delle firme corrisponde ad una Nuova Firma

Soluzione: Funzione Hash

Proprietà funzione hash

- comprime il messaggio input
- calcolabile efficientemente
- computazionalmente difficile trovare 2 diversi messaggi con lo stesso valore hash
- dato y è computazionalmente difficile trovare M tale che y = h(M)

Firma digitale con hash

- Invece di firmare il messaggio M firmiamo il messaggio di dimensione fissata hash(M)
- **Vantaggi**
  - efficienza
  - sicurezza
  - integrità

Existential Forgery/ Key-only attack
L'attaccante riesce a generare una firma (F, scelta a caso) per un valore $z=F^e$
L'avversario non potrà calcolare un messaggio M per cui z = hash(M)

Embedding

- Uno schema di embedding permette di mappare un messaggio da firmare in una rappresentazione del messaggio (stringa) che sarà effettivamente firmata

- Gli schemi di embedding servono per evitare possibili attacchi allo schema di firma (tipo existential forgery)

Proprietà funzione di embedding

- La funzione μ dovrebbe comportarsi come una funzione hash. Deve soddisfare le proprietà di
- one-way
  - per un x a caso è difficile trovare un messaggio M tale che $x = \mu(M)$
- collision-resistant
  - è difficile trovare due messaggi M_1 ed M_2 tali che $\mu(M_1)=\mu(M_2)$

Embedding per firma digitale con appendice

- ANSI X9.31
- PKCS #1 v1.5
- Bellare-Rogaway
  - FDH (Full Domain Hashing)
  - PSS (Probabilistic Signature Scheme)
- IEEE P1363a
  - Include varianti di BR-FDS e BR-PSS

Schemi con recupero del messaggio

Anche $\mu(M) = Hash(M)$ non funziona
Existential (Multiplicative) Forgery

- Schema RSA di base
  - $\mu(M) = M$, insicuro - non dovrebbe essere usato
- ISO/IEC 9796-1
  - **Rotto**
- ISO/IEC 9796-2, prima versione nel 2002
  - **Rotto** (Aggiornato nel 2010, confermato nel 2016)
- Bellare-Rogaway PSS-R
- IEEE P1363a si basa su PSS-R

Attacchi

ISO/IEC 9796-1

D. Coppersmith, S. Halevi and C. Jutla,
ISO 9796-1 and the new, forgery strategy
Rump Session CRYPTO 1999, 1999

Francois Grieu
A Chosen Messages Attack on the ISO/IEC 9796–1 Signature Scheme
LNCS 1807, EUROCRYPT 2000, pp 70-8

Rotto con solo 3 messaggi scelti

D. Coppersmith, J.S. Coron, F. Grieu, S. Halevi, C. Jutla, D. Naccache, and J.P. Stern
Cryptanalysis of ISO/IEC 9796-1
Journal of Cryptology, 2008

ISO/IEC 9796-2

J.S. Coron, D. Naccache, M. Tibouchi and R.P. Weinmann
Practical Cryptanalysis of ISO 9796-2 and EMV Signatures
Journal of Cryptology, 2016

ANSI X9.31

- μ(M) =
  0x6b 0xbb ... 0xbb 0xba || Hash(M) || 0x3y 0xcc
  - y = 3 per SHA-1
  - y = 1 per RIPEMD-160
- Ampiamente standardizzato
  - IEEE 1363, ISO/IEC 14888-3
  - US NIST FIPS 186-1
- ANSI X9.31 richiede che p e q siano *strong prime*

EMSA (PKCS#1)

- Si tratta di un *encoding method for signatures with appendix* per la firma RSA descritto nello standard PKCS#1
- Lo standard PKCS#1 descrive due metodi
  - EMSA-PKCS1-v1_5 (descritto in PKCS#1v1.5)
    - embedding deterministico, conservato per compatibilità con applicazioni esistenti. Non ci sono prove di sicurezza, né attacchi
  - EMSA-PSS (descritto in PKCS#1v2.2)
    - embedding randomizzato, è provato sicuro, da usare per nuove applicazioni

M. Bellare and P. Rogaway.
The Exact Security of Digital Signatures - How to Sign with RSA and Rabin, Eurocrypt 1996, LNCS 1070,1996

EMSA-PKCS1-v1_5

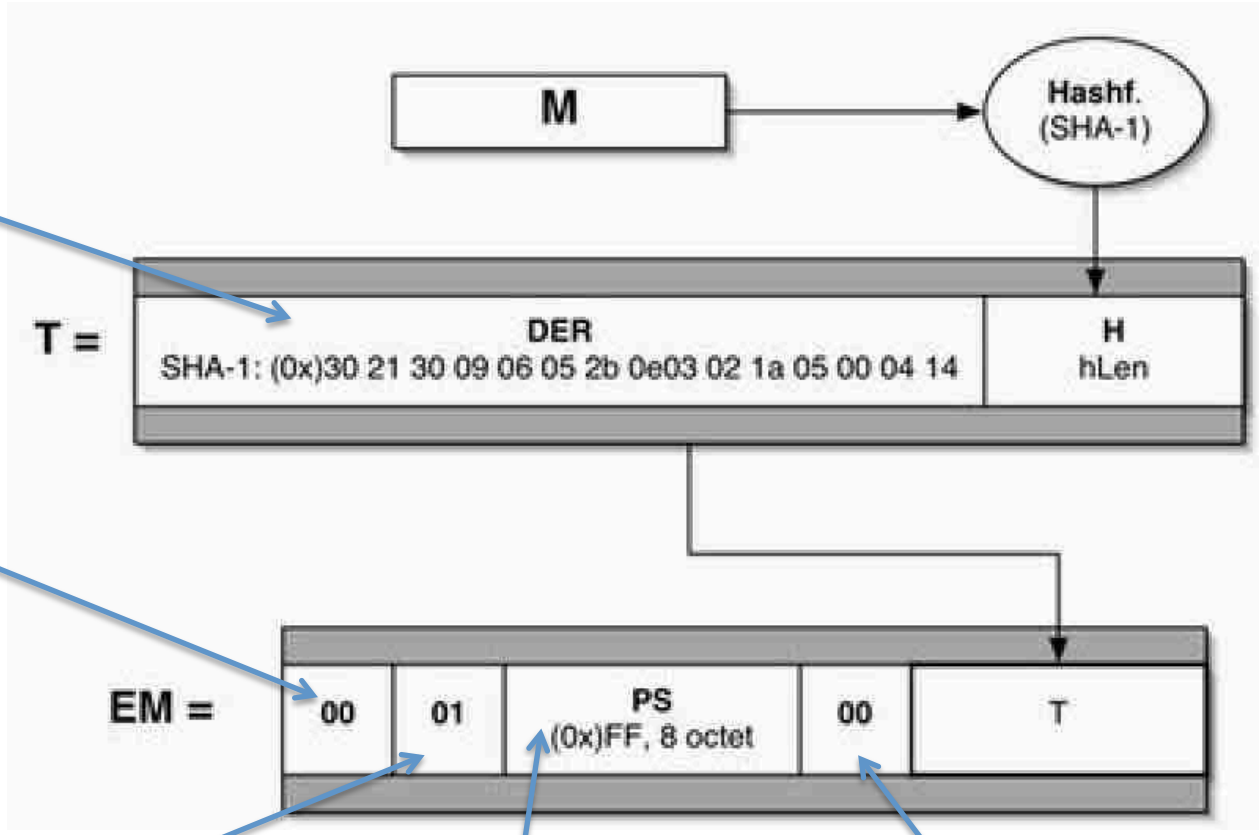
Codifica l'identificativo della funzione hash

Assicura che EM sia in Z_n

Indica che si tratta di una firma

padding string di almeno 8 byte

Fine padding



Identificativo funzione hash

in grassetto

MD2:	**(0x)30**	**20**	**30**	**0c**	**06**	**08**	**2a**	**86**	**48**	**86**	**f7**	**0d**	**02**	**02**	**05**	**00**	**04**	**10**	|| *H.*
MD5:	**(0x)30**	**20**	**30**	**0c**	**06**	**08**	**2a**	**86**	**48**	**86**	**f7**	**0d**	**02**	**05**	**05**	**00**	**04**	**10**	|| *H.*
SHA-1:	**(0x)30**	**21**	**30**	**09**	**06**	**05**	**2b**	**0e**	**03**	**02**	**1a**	**05**	**00**	**04**	**14**	|| *H.*			
SHA-224:	**(0x)30**	**2d**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**04**	**05**	**00**	**04**	**1c**	|| *H.*
SHA-256:	**(0x)30**	**31**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**01**	**05**	**00**	**04**	**20**	|| *H.*
SHA-384:	**(0x)30**	**41**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**02**	**05**	**00**	**04**	**30**	|| *H.*
SHA-512:	**(0x)30**	**51**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**03**	**05**	**00**	**04**	**40**	|| *H.*
SHA-512/224:	**(0x)30**	**2d**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**05**	**05**	**00**	**04**	**1c**	|| *H.*
SHA-512/256:	**(0x)30**	**31**	**30**	**0d**	**06**	**09**	**60**	**86**	**48**	**01**	**65**	**03**	**04**	**02**	**06**	**05**	**00**	**04**	**20**	|| *H.*

Definiti in PKCS#1v1.5, presenti anche in PKCS#1v2.2

EMSA-PSS

$$s = \mathrm{sig}_{k_{pr}}(x) \equiv EM^d \bmod n$$

EMSA per Probabilistic Signature Scheme

Let $|n|$ be the size of the RSA modulus in bits. The encoded message EM has a length $\lceil (|n| - 1)/8 \rceil$ bytes such that the bit length of EM is at most $|n| - 1$ bit.

1. Generate a random value *salt*.
2. Form a string M' by concatenating a fixed padding $padding_1$, the hash value $mHash = h(M)$ and *salt*.
3. Compute a hash value H of the string M'.
4. Concatenate a fixed padding $padding_2$ and the value *salt* to form a data block *DB*.
5. Apply a mask generation function *MGF* to the string M' to compute the mask value *dbMask*. In practice, a hash function such as SHA-1 is often used as *MGF*.
6. XOR the mask value *dbMask* and the data block *DB* to compute *maskedDB*.
7. The encoded message *EM* is obtained by concatenating *maskedDB*, the hash value H and the fixed padding *bc*.

EMSA-PSS

salt: valore casuale di sLen byte

Defaul: sLen=20

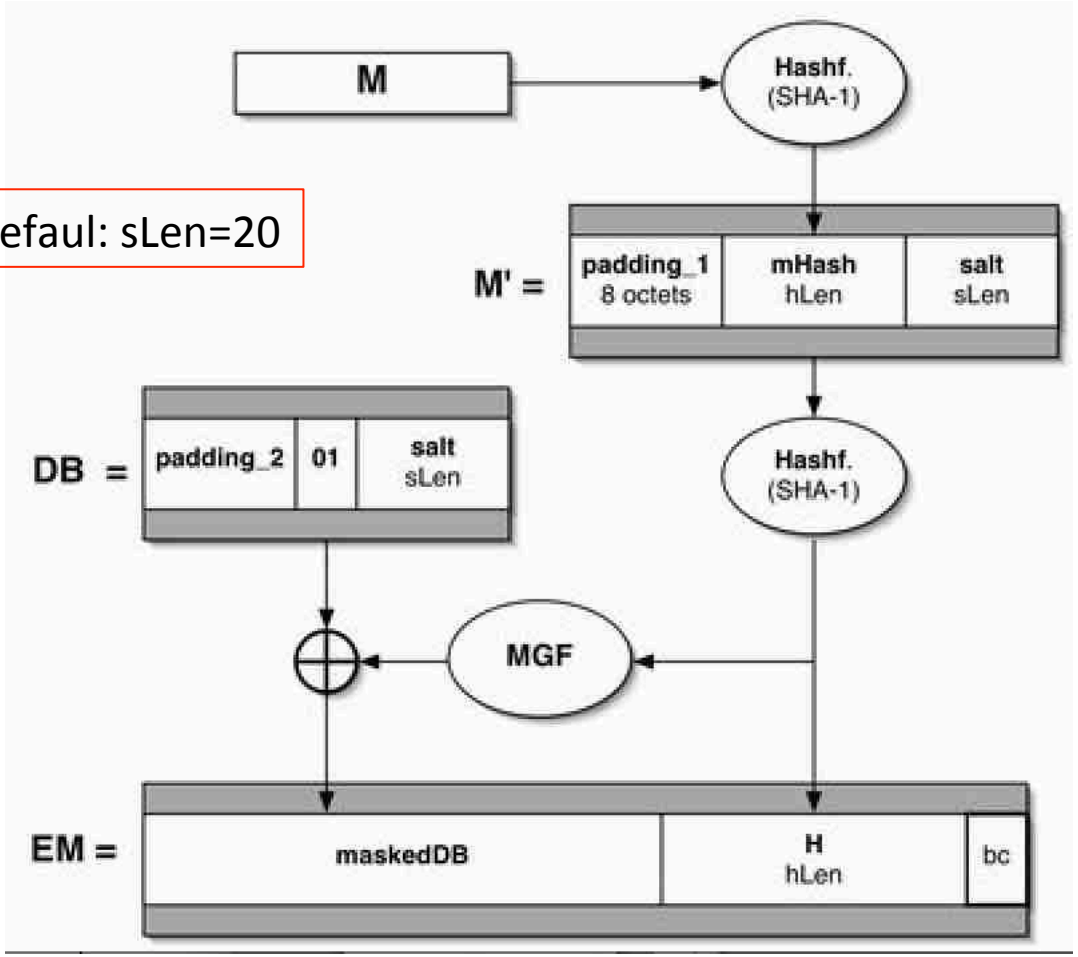
$padding_1$: otto ottetti di valore 0x00

emLen: lunghezza in byte di EM

01: valore fissato pari a 0x01

$padding_2$: emLen- sLen - hLen- 2 ottetti di valore 0x00

MGF: Mask Generation Function



bc: valore fissato pari a 0xbc

Se |n| è la dimensione in bit del modulo RSA, il messaggio codificato EM sarà lungo $\lceil (|n| - 1)/8 \rceil$ byte in modo che la lunghezza in bit di EM è al più |n| – 1 bit

Firma digitale El Gamal

Firma digitale El Gamal

- Sicurezza basata sull'intrattabilità del problema del logaritmo discreto
- Si basa su Z_p^* o su di un suo sottogruppo
  - è necessario un generatore del gruppo

Key Generation for Elgamal Digital Signature

1. Choose a large prime p.
2. Choose a primitive element α of $\mathbb{Z}_p^*$ or a subgroup of $\mathbb{Z}_p^*$.
3. Choose a random integer $d \in \{2, 3, \ldots, p-2\}$.
4. Compute $\beta = \alpha^d \bmod p$.

$$k_{pub} = (p, \alpha, \beta)$$

$$k_{pr} = d$$

Firma/Verifica El Gamal

Elgamal Signature Generation

1. Choose a random ephemeral key $k_E \in \{2, \ldots, p-2\}$ such that $\gcd(k_E, p-1) = 1$.
2. Compute the signature parameters:

$$r \equiv \alpha^{k_E} \bmod p,$$
$$s \equiv (x - d \cdot r)\, k_E^{-1} \bmod p - 1.$$

La firma è (r,s)

Elgamal Signature Verification

1. Compute the value

$$t \equiv \beta^r \cdot r^s \bmod p$$

2. The verification follows from:

$$t \begin{cases} \equiv \alpha^x \bmod p & \Longrightarrow \text{valid signature} \\ \not\equiv \alpha^x \bmod p & \Longrightarrow \text{invalid signature} \end{cases}$$

r corrisponde alla chiave effimera del cifrario El Gamal

Perché funziona?

$$r \equiv \alpha^{k_E} \bmod p,$$
$$s \equiv (x - d \cdot r)\, k_E^{-1} \bmod p - 1$$

Calcoliamo

$$\begin{aligned} \beta^r \cdot r^s &\equiv (\alpha^d)^r (\alpha^{k_E})^s \bmod p \\ &\equiv \alpha^{d\,r + k_E\,s} \bmod p. \end{aligned}$$

La firma è valida se $\alpha^x \equiv \beta^r \cdot r^s \mod p$ quindi se

$$\alpha^x \equiv \alpha^{d\,r + k_E\,s} \bmod p$$

Secondo il Piccolo Teorema di Fermat, l'espressione è soddisfatta se

$x \equiv d\,r + k_E\,s \bmod p - 1$ quindi se $s \equiv (x - d \cdot r) k_E^{-1} \bmod p - 1$

che corrisponde al calcolo del valore s

Aspetti Computazionali

- Generazione chiave
  - Identica a quella per lo schema di cifratura El Gamal
  - p deve essere di almeno 1024 bit
- Firma
  - Dimensione firma tre volte dimensione messaggio
  - Un elevamento a potenza + una moltiplicazione + il calcolo di un inverso modulare
- Verifica
  - Due elevamenti a potenza + una moltiplicazione modulare

Tuple (r, k_E, k_E^{-1}) possono essere calcolate in anticipo, memorizzate e utilizzate quando necessario

Sicurezza

- La sicurezza dello schema di firma El Gamal si basa sulla difficoltà di risolvere DLP (Discrete Logarithm Problem)
- Risolvendo DLP si può estrarre d da β
- Bisogna scegliere i parametri in maniera tale che DLP si difficile da risolvere
  - Si sceglie un generatore di un sottogruppo di $\mathbb{Z}_p^*$ di ordine primo. Il sottogruppo deve essere di grande dimensione

Riuso della chiave effimera k_E

- Assumiamo che due messaggi x_1 ed x_2 siano firmati utilizzando la stessa chiave effimera k_E
  - L'attaccante se ne accorge perché r non cambia
  - (r_1, s_1) firma di x_1 e (r_2, s_2) firma di x_2 con

$$r_1 = r_2 = \alpha^{k_E} \qquad \text{e} \qquad \begin{aligned} s_1 &\equiv (x_1 - d\,r)k_E^{-1} \bmod p-1 \quad \text{(a)} \\ s_2 &\equiv (x_2 - d\,r)k_E^{-1} \bmod p-1 \quad \text{(b)} \end{aligned}$$

da cui $$s_1 - s_2 \equiv (x_1 - x_2)k_E^{-1} \bmod p-1$$

Si ricava $$k_E \equiv \frac{x_1 - x_2}{s_1 - s_2} \bmod p-1$$ Se $mcd(s_1\text{-}s_2, p\text{-}1) = 1$

Da k_E e dalla (a) si calcola d come $$d \equiv \frac{x_1 - s_1 k_E}{r} \bmod p-1$$

Existential Forgery

Alice **Oscar** **Bob**

$k_{pr} = d$

$k_{pub} = (p, \alpha, \beta)$

$\xleftarrow{(p,\alpha,\beta)}$ $\xleftarrow{(p,\alpha,\beta)}$

1. select integers i, j
   where $\gcd(j, p-1) = 1$
2. compute signature:
   $r \equiv \alpha^i \beta^j \bmod p$
   $s \equiv -r\, j^{-1} \bmod p - 1$

3. compute message:
   $x \equiv s\, i \bmod p - 1$

$\xleftarrow{(x,(r,s))}$

verification:
$t \equiv \beta^r \cdot r^s \bmod p$
since $t \equiv \alpha^x \bmod p$:
valid signature!

Verifica superata?

$$
\begin{aligned}
t &\equiv \beta^r \cdot r^s \bmod p \\
&\equiv \alpha^{dr} \cdot r^s \bmod p \\
&\equiv \alpha^{dr} \cdot \alpha^{(i+dj)s} \bmod p \\
&\equiv \alpha^{dr} \cdot \alpha^{(i+dj)(-rj^{-1})} \bmod p \\
&\equiv \alpha^{dr-dr} \cdot \alpha^{-rij^{-1}} \bmod p \\
&\equiv \alpha^{si} \bmod p
\end{aligned}
$$

Poiché $x \equiv si \bmod p-1$

abbiamo $\alpha^{si} \equiv \alpha^x \bmod p$

quindi $t \equiv \alpha^x \bmod p$

Possiamo evitare l'attacco firmando, al posto del messaggio x, il valore hash del messaggio

$$s \equiv (h(x) - d \cdot r)k_E^{-1} \bmod p-1.$$

Riferimenti

Christof Paar and Jan Pelzl
Understanding Cryptography
Capitolo 10 Digital Signatures
Paragrafi 1, 2 e 3