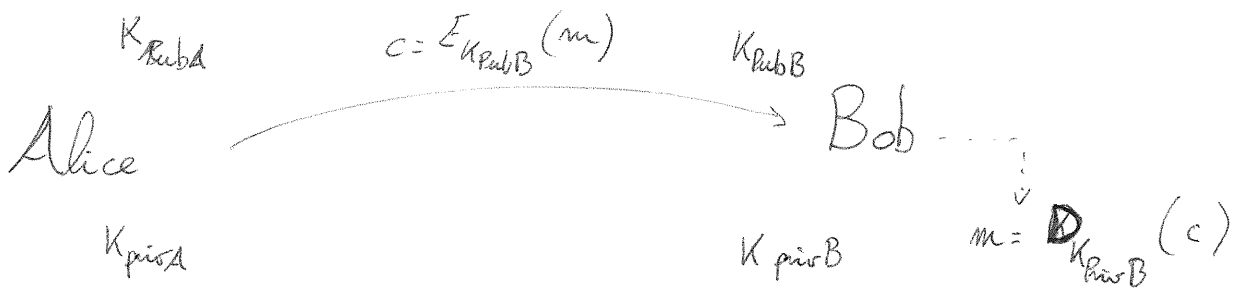
Crittografia a chiave pubblica

Scenario



Alice usa K_{PubB} per cifrare il messaggio m per Bob.

Bob usa K_{PrivB} per decifrare il messaggio c ricevuto.

Definizione 11.1. Uno schema di cifratura a chiave pubblica è una terna di algoritmi di tempo polinomiale probabilistici (Gen, Enc, Dec) tali che:

1. Gen prende in input il parametro di sicurezza e dà in output (pK, sK) → chiave privata, chiave pubblica

2. Enc prende in input una chiave pubblica pk ed un messaggio dallo spazio dei messaggi (che può dipendere da pk)
   Dà in output il cifrato c, i.e., $c \leftarrow Enc(pk, m)$

3. Dec prende in input una chiave privata sk ed un cifrato c e dà in output un messaggio m o $\perp$ (che denota errore), i.e., $m := Dec(sk, c)$

↑ deterministico

$(pk, sk) \leftarrow Gen(1^n)$; $c \leftarrow Enc_{pk}(m)$; $m := Dec_{sk}(c)$

notazione usuale

È richiesto che, eccetto con probabilità trascurabile sulla scelta di (pk, sk) effettuata da $Gen(1^n)$, risulta

$$Dec_{sk}(Enc_{pk}(m)) = m$$

per ogni messaggio (legale) m.

Osservazioni: comparazione con gli schemi simmetrici.

- ricevenți multipli: nessuna chiave da memorizzare
- risolto il problema della condivisione iniziale
- chiave segreta mai "esposta"
- performance peggiori

Sicurezza rispetto ad attacchi CPA

Possiamo rivisitare le definizioni fornite nel contesto simmetrico ed adattarle allo scenario a chiave pubblica.

Cominciamo con l'esperimento* dell'indistinguibilità rispetto ad un avversario che ascolta

$\text{PubK}^{\text{eav}}_{A,\Pi}(n)$ $\quad \Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, A adv, n par. sic.

1. $(PK, SK) \leftarrow \text{Gen}(1^n)$; (Eseguito da C)
2. $A(PK)$ dà in output m_0, m_1, appartenenti allo spazio dei messaggi, tali che $|m_0| = |m_1|$
3. C sceglie uniformemente $b \in \{0,1\}$, calcola $c \leftarrow \text{Enc}_{PK}(m_b)$ e lo dà ad A.
   ↳ (cifrato di sfida)
4. A dà in output b'. L'output dell'esperimento è 1 se $b' = b$, è 0 altrimenti.
   ↳ (A ha successo)

* Immaginiamo sempre che l'esperimento sia gestito da un Challenger C con cui A interagisce.

Definizione 11.2 Una schema di cifratura a chiave pubblica $\Pi = (Gen, Enc, Dec)$ ha cifrature indistinguibili in presenza di un avversario da ascolto se, per ogni A PPT, esiste una funzione trascurabile negl tale che

$$Pr\left[Priv^{eav}_{A,\Pi}(n) = 1 \right] \leq \frac{1}{2} + negl(n)$$

A non deve essere in grado di individuare m_b con probabilità significativamente migliore (i.e., non trascurabile) di $1/2$.

Osservazione: rispetto all'esperimento $PrivK^{eav}_{A,\Pi}$, l'avversario

A RICEVE la chiave pubblica PK.

$\Rightarrow$ equivale ad aver accesso ad un oracolo per la cifratura ($PrivK^{cpa}_{A,\Pi}$)

$\Rightarrow$ la definizione 11.2 <u>coincide</u> con quella di sicurezza rispetto ad attacchi CPA!

Proposizione 11.3 Se uno schema di cifratura a chiave pubblica ha cifrature indistinguibili in presenza di un avversario che ascolta, allora è CPA-sicuro.

Nota: ciò contrasta con lo scenario simmetrico.

Abbiamo visto che esistono schemi che

- hanno cifrature indistinguibili
- sono insicuri rispetto ad attacchi CPA

Osservazioni: Schemi di cifratura a chiave pubblica perfettamente sicuri sono IMPOSSIBILI.

Infatti, ogni avversario A con potenza di calcolo illimitata, data PK ed un cifrato c, può determinare m con probabilità 1.

Così come abbiamo evidenziato nel contesto simmetrico, schemi di cifratura DETERMINISTICI non possono essere CPA sicuri.

Teorema 11.4 Nessuno schema di cifratura a chiave pubblica deterministico ha cifrature indistinguibili in presenza di un avversario che ascolta, i.e., è CPA-sicuro.

Per convincersi del risultato, si noti che:

- un avversario A, rispetto ad uno schema deterministico, ha sempre la capacità di stabilire se un messaggio è stato inviato due volte (i.e., stesso cifrato)
- se lo spazio dei messaggi da cifrare è piccolo, può sempre recuperare m da c con probabilità 1.

Il lavoro di Goldwasser e Micali, che introdusse la Definizione 11.2 ed enunciò il Teorema 11.4, ha rappresentato un punto di svolta in crittografia, mostrando l'importanza della cifratura probabilistica.

Cifratura di messaggi di lunghezza arbitraria

Una conseguenza immediata del teorema precedente è che uno schema di cifratura a chiave pubblica CPA-sicuro per messaggi di lunghezza fissa può essere usato per costruirne uno CPA-sicuro per messaggi di lunghezza arbitraria.

$\Pi = (Gen, Enc, Dec)$ → messaggi di 1 bit

$\Pi' = (Gen', Enc', Dec')$ → messaggi di lungh. arb.
i.e., $m \in \{0,1\}^*$

$Gen' \equiv Gen$

$$Enc'_{PK}(m) = Enc_{PK}(m_1)\, Enc_{PK}(m_2) \cdots Enc_{PK}(m_\ell)$$

dove $m = m_1 m_2 \cdots m_\ell$ (ogni m_i: 1 bit)

$$Dec'_{SK}(c) = Dec_{SK}(c_1)\, Dec_{SK}(c_2) \cdots Dec_{SK}(c_\ell)$$

(ogni output: 1 bit)

Sicurezza rispetto ad attacchi di tipo CCA.

A riesce ad ottenere la decifratura di cifrati arbitrari di scelta propria

Nell'esperimento è modella l'attacco equivale ad un accesso ad un oracolo di decifratura

In diversi scenari concreti, attacchi di questo tipo possono essere una minaccia reale.

- $\text{PrivK}^{\text{cca}}_{A,\Pi}(n)$ A adv, $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, n par. sic.

1. $(PK, SK) \leftarrow \text{Gen}(1^n)$ (eseguito da C)
2. $A(PK)$ riceve accesso oracolare a $\text{Dec}_{SK}(\cdot)$. Dà in output una coppia (m_0, m_1) tale che $|m_0| = |m_1|$ (appartenenti allo spazio dei messaggi)
3. C sceglie un bit $b \in \{0,1\}$ in modo uniforme, calcola $c \leftarrow \text{Enc}_{PK}(m_b)$ e lo dà a A
4. A continua ad interagire con l'oracolo $\text{Dec}_{SK}(\cdot)$ ma non può chiedere di decifrare c.

Infine dà in output b'.

5. L'output dell'esperimento è 1 se $b = b'$, e invece 0 altrimenti.

Definizione 11.8 Uno schema di cifratura a chiave pubblica $\Pi = (Gen, Enc, Dec)$ ha cifrature indistinguibili rispetto ad attacchi di tipo chosen ciphertext (o è CCA-sicuro) se, per ogni avversario A PPT, esiste una funzione trascurabile $negl$, tale che

$$Pr\left[PubK^{cca}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl$$

Fortunatamente, per la nozione di sicurezza CCA vale un analogo del Teorema 11.6.

- Se uno schema ha cifrature indistinguibili rispetto ad un attacco di tipo CCA, allora ha anche cifrature <u>multiple</u> indistinguibili rispetto ad un attacco di tipo CCA.

Osservazione: Nota che la tecnica utilizzata per costruire uno schema di cifratura Π' a partire da uno schema di cifratura Π per messaggi di lunghezza fissata, al fine di ottenere uno schema per messaggi di lunghezza arbitraria, non funziona per schemi CCA-sicuri

Π CPA-sicuro $\Rightarrow$ Π' CPA-sicuro

Π CCA-sicuro $\not\Rightarrow$ Π' CCA-sicuro

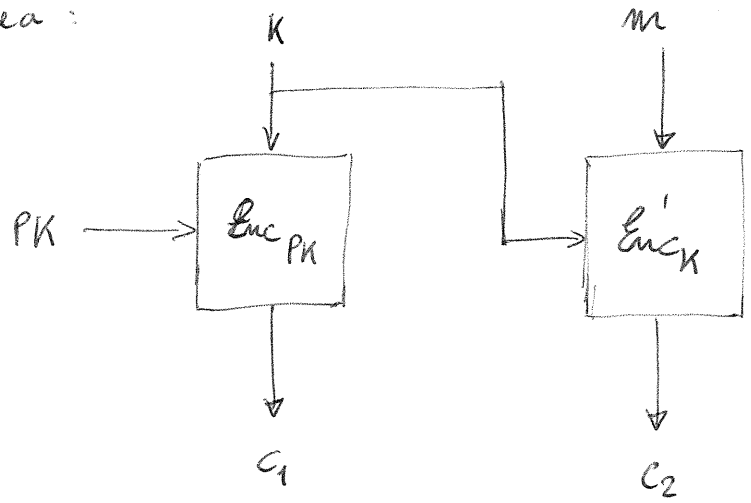
Cifrari Ibridi

Utilizzano cifratura a chiave pubblica e cifratura simmetrica.

Migliorano l'efficienza d'uso di uno schema a chiave pubblica

- gli schemi simmetrici sono molto più veloci nelle operazioni di cifratura/decifratura

Idea:



Enc_{PK} = cifratura a chiave pubblica

Enc'_K = cifratura simmetrica

Nell'approccio di base, Enc_{PK} cifra la chiave K scelta dal mittente uniformemente a caso, e poi Enc'_K cifra con K il messaggio m.

Approcci più diretti per "racchiudere" tutto in un passo sono detti KEM - meccanismi di incapsulamento della chiave

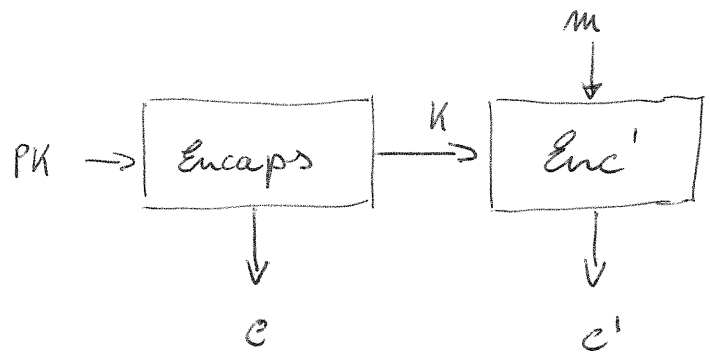
Essenzialmente un KEM ha tre algoritmi: Gen per generare la coppia di chiavi (PK, SK), Encaps che prende in input PK e dà in output (c, K)
↳ chiave simmetrica
e Decaps che prende in input SK e c e dà in output K o $\perp$ (messaggio d'errore).

Definizione 11.9 Un meccanismo per l'incapsulamento della chiave (KEM, key-encapsulation mechanism) è una tripla di algoritmi PPT (Gen, Encaps, Decaps) tali che:

1. $Gen(1^n) \rightarrow (PK, SK)$ alg. generazione chiavi
2. $Encaps(PK, 1^n) \rightarrow (c, K)$ $\quad K \in \{0,1\}^{\ell(n)}$
   algoritmo di incapsulamento della chiave
3. $Decaps(SK, c) \rightarrow K$ $\quad (K := Decaps(SK, c))$
   ↓ deterministico

Ogni schema di cifratura a chiave pubblica può essere usato per costruire un KEM, scegliendo K uniformemente a caso e cifrandola con PK.

Usando un KEM possiamo implementare uno schema di cifratura ibrido come segue



In questo contesto, Enc' viene detto un "meccanismo per l'incapsulamento dei dati" (DEM, data-encapsulation mechanism)

Costruzioni dedicate di KEM possono essere più efficienti rispetto all'approccio di base.

Alice, per inviare un messaggio cifrato a Bob

- usa $Encaps(PK, 1^n)$ ed ottiene (c, K)
  - PK: chiave pubblica di Bob
  - K: chiave simmetrica
- calcola $c' = Enc'_K(m)$ ed invia (c, c') a Bob.

Bob decifra il messaggio (c, c') usando

- $Decaps(SK, c) = K$ e $Dec_K(c') = m$
  - SK: propria chiave privata
  - K: chiave simmetrica

Costruzioni basate sulle assunzioni CDH / DDH

Cifratura El Gamal

Proposto nel 1985 da Taher El Gamal, è una semplice modifica dello scambio di chiavi Diffie-Hellman.

Intuizione: Sia K la chiave stabilita tra Alice e Bob e sia m il messaggio che Alice vuole inviare a Bob



$$m = \frac{c}{K}$$

Se K è indistinguibile da una chiave casuale allora anche il cifrato c è indistinguibile da un valore casuale e non fornisce alcuna informazione su m.

In modo più formale, procediamo come segue:

Sia $\mathcal{G}$ un algoritmo PPT che su input 1^n dà in output la descrizione di un gruppo ciclico G, di ordine q (con $\|q\| = n$), ed un generatore g.

- Costruzione 11.16

$\Pi = (Gen, Enc, Dec)$ è definito come segue

- Gen: $(G, q, g) \leftarrow \mathcal{G}(1^n)$;
  scegli uniformemente $x \in \mathbb{Z}_q$, calcola $h := g^x$
  Chiave pubblica: $\langle G, q, g, h \rangle$
  Chiave privata: $\langle G, q, g, x \rangle$
  Spazio dei messaggi: G

- Enc: su input $\langle G, q, g, h \rangle$ ed $m \in G$,
  scegli uniformemente $y \in \mathbb{Z}_q$ e calcola

$$\langle g^y, h^y \cdot m \rangle$$

- Dec: su input $\langle G, q, g, x \rangle$ e $\langle c_1, c_2 \rangle$,
  dà in output

$$\hat{m} := c_2 / c_1^x .$$

Schema di cifratura El Gamal

È facile vedere che lo schema è corretto. Dati

$$\langle c_1, c_2 \rangle = \langle g^y, h^y \cdot m \rangle \quad \text{con } h = g^x$$

risulta:

$$\hat{m} = \frac{c_2}{c_1^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{(g^x)^y \cdot m}{g^{xy}} = \frac{g^{xy} \cdot m}{g^{xy}} = m$$

Un esempio numerico:

Sia $q = 83$, $p = 2q + 1 = 167$ e sia G il gruppo dei residui quadratici mod p. (p e q primi $\Rightarrow$ G è un sottogruppo di $\mathbb{Z}_p^*$ di ordine q)

$|G| = q$ primo $\Rightarrow$ ogni elemento $\neq 1$ è un generatore.

Prendiamo $g = 2^2 = 4 \bmod 167$. Il ricevente sceglie come chiave segreta $37 \in \mathbb{Z}_{83}$ e la chiave pubblica è

"rappresenta G - l'utente su cui..."

$$PK = \langle p, q, g, h \rangle = \langle 167, 83, 4, [4^{37} \bmod 167] \rangle = \langle 167, 83, 4, 76 \rangle$$

Un mittente che vuole inviare $m = 65 \in G$ ($65 = 30^2 \bmod 167$) sceglie y, diciamo $y = 71$, e calcola

$$(\langle 4^{71} \bmod 167,\ 76^{71} \cdot 65 \bmod 167 \rangle) = \langle 132, 44 \rangle$$

Per decifrare, il ricevente, prima calcola $12 = [132^{37} \bmod 167]$. Poi, essendo $66 = [12^{-1} \bmod 167]$, $m = 65 = [44 \cdot 66 \bmod 167]$.

Teorema 11.18. Se il problema DDH è difficile relativamente a $\mathcal{G}$, allora lo schema di cifratura El Gamal è CPA-sicuro.

Alcune considerazioni sugli aspetti implementativi

- Condivisione dei parametri pubblici.

  Fissati e condivisi generalmente in anticipo.

  Il NIST ha pubblicato un insieme di parametri consigliati (per (G, q, g)).

- Scelta del gruppo.

  Solitamente q è un primo.

  Gruppi frequentemente utilizzati: gruppi basati su punti di curve ellittiche, sottogruppi di $\mathbb{Z}_p^*$ (p primo)

- Spazio dei messaggi (G è "scomodo")

  Usare $\{0,1\}^e$ e poi servirsi di una codifica reversibile per passare ad elementi di G.

Un'idea alternativa è invece usare ElGamal per produrre uno schema ibrido:

- il sender può scegliere $z \in G$, lo cifra con ElGamal, e lo manda al ricevente
- usa $H: G \to \{0,1\}^n$ per calcolare $H(z) = K$
- cifra m con un cifrario simmetrico usando K.

H viene detta "funzione di derivazione" della chiave.

Ma esistono approcci ibridi più efficienti.

- Costruzione 11.19

Sia $\mathcal{G}$ un generatore di gruppi ciclici. Definiamo un KEM come segue:

- Gen: su input 1^n, $(G, q, g) \leftarrow \mathcal{G}(1^n)$;
  scegli $x \leftarrow \mathbb{Z}_q$, pone $h := g^x$.
  specifica $H: G \to \{0,1\}^{\ell(n)}$.
  $PK = \langle G, q, g, h, H \rangle$, $SK = \langle G, q, g, x \rangle$
- Encaps: su input $PK = \langle G, q, g, h, H \rangle$, scegli $y \in \mathbb{Z}_q$ e dà in output g^y e la chiave $H(h^y)$
- Decaps: su input $SK = \langle G, q, g, x \rangle$ e $c \in G$, dà in output la chiave $H(c^x)$.

Teorema 11.20 Se il problema DDH è difficile relativamente a $\mathcal{G}$, ed H è "scelta in modo opportuno", allora la Costruzione 11.19 genera un KEM CPA-sicuro.

Come scegliere H? $H: G \to \{0,1\}^\ell$

- "vicina" a quella uniforme: per ogni $K \in \{0,1\}^\ell$ il # elementi di G che producono K è approssimativamente lo stesso
- H è una funzione con chiave (dove la chiave è inclusa nella descrizione della chiave pubblica del ricevente) che si comporta come un "estrattore forte".
- H è un oracolo casuale: in quest'ultimo caso la Costruzione 11.19 può essere provata CPA-sicura nel modello ROM, basandosi sulla più debole assunzione CDH.

Teorema 11.21. Se il problema CDH è difficile rel. a $\mathcal{G}$, e H è un oracolo casuale, allora la Costruzione 11.19 è CPA-sicura.

Sicurezza CCA

È facile vedere che El Gamal è vulnerabile ad attacchi di tipo chosen-ciphertext. Infatti, è malleabile!

A intercetta $c = \langle c_1, c_2 \rangle$ cifrato da $pk = \langle G, q, g, h \rangle$

costruisce $c' = \langle c_1, c_2' \rangle$ dove $c_2' = c_2 \cdot \alpha$, $\alpha \in G$

$\Rightarrow$ se c è una cifratura di m, cioè

$$c_1 = g^y \quad e \quad c_2 = h^y \cdot m \quad (y \in \mathbb{Z}_q),$$

allora risulta $c_1 = g^y$ e

$c_2' = h^y \cdot (\alpha \cdot m)$ ovvero c' è una cifratura di $\alpha \cdot m$.

Quindi A può trasformare la cifratura di un messaggio (sconosciuto) m nella cifratura di un messaggio (sconosciuto) $\alpha \cdot m$ (α, conosciuta)

Lo schema KEM precedente può essere malleabile. Dipende dalla scelta della funzione H.

Se H è modellata come un oracolo casuale, si può dimostrare che il KEM è CCA-sicuro. Precisamente:

In precedenza abbiamo dimostrato che

Π è un KEM CCA-sicuro
Π' è uno schema di cifratura simmetrico CCA-sicuro
$\Rightarrow$ lo schema ibrido Π^{hy} è CCA-sicuro

Pertanto possiamo ottenere uno schema di cifratura ibrido a chiave pubblica CCA-sicuro usando:

- KEM della costruzione precedente
- Uno schema simmetrico di cifratura autenticata (basato per esempio sul paradigma "cifra e poi autentica")

Lo schema risultante corrisponde a quanto fatto negli schemi DHIES/ECIES, che sono inclusi nello standard ISO/IEC 18033-2 per la cifratura a chiave pubblica.

La cifratura in questi schemi prende la forma

$$\langle g^Y, Enc'_{k_E}(m), Mac_{k_M}(c') \rangle$$

dove $c' = Enc'_{k_E}(m)$ ed Enc' è CPA-sicuro.

Precisamente:

DHIES (Diffie-Hellman Integrated Encryption Scheme) può essere usato genericamente per indicare uno schema di questa forma, o per indicare il caso in cui il gruppo

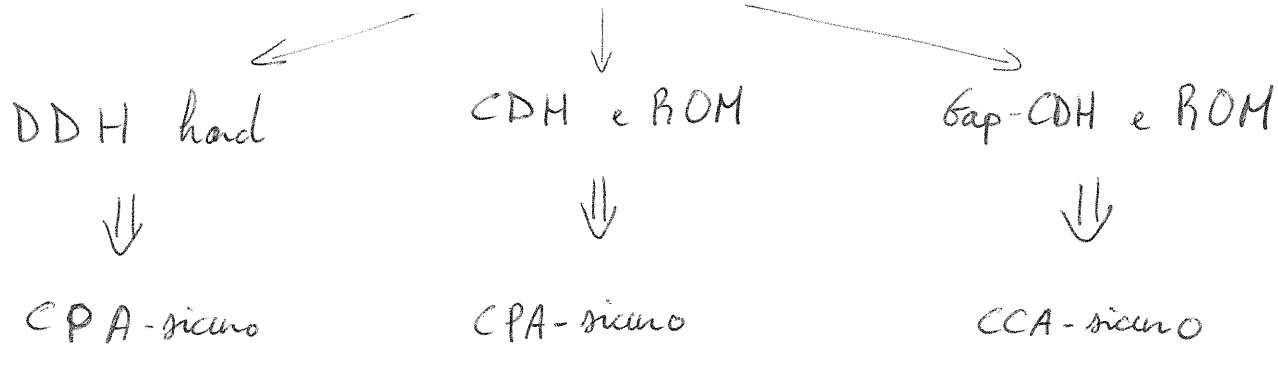
G è un sottogruppo ciclico di un campo finito.

ECIES (Elliptic Curve Integrated Encryption Scheme)

G è un gruppo di punti di una curva ellittica.

Corollario 11.24. Se Π_E è CPA-sicuro, Π_M è un MAC fortemente sicuro, H è un oracolo casuale e il problema Gap-CDH è difficile relativamente a $\mathcal{G}$, allora la Costruzione 11.23 (DHIES/ECIES) è CCA-sicura.

Nota: il KEM "ElGamal-like" di Costruzione 11.19



L'assunzione della Fattorizzazione

GenModulus, algoritmo di tempo polinomiale, tale che

↳ input: 1^n

output: (N, p, q), $N = p \cdot q$, p, q primi di n bit

(eccetto con probabilità trascurabile)

Modo naturale: calcola primi p e q, moltiplica ...

Esperimento $\text{Factor}_{A, \text{GenModulus}}(n)$

1. Esegui $\text{GenModulus}(1^n)$ per ottenere (N, p, q)
2. A riceve N, e dà in output $p', q' > 1$
3. L'output dell'esperimento è 1 se $p' \cdot q' = N$, 0 altrimenti.

Nota che se l'output è 1, allora $\{p', q'\} = \{p, q\}$ a meno che p e q siano composti (prob. trascurabile)

Definizione 8.45 La fattorizzazione è difficile relativamente a GenModulus se, $\forall A$ PPT, $\exists$ una funzione negl trascurabile, tale che

$$\Pr\left[\text{Factor}_{A, \text{GenModulus}}(n) = 1\right] \leq \text{negl}(n)$$

L'assunzione della Fattorizzazione $\iff$ $\exists$ GenModulus relativamente al quale la fattorizzazione è difficile.

L'assunzione RSA

GenRSA, algoritmo di tempo polinomiale, tale che

↳ input: 1^n

output: un modulo N, prodotto di due primi di n bit ed interi $e, d > 0$, con $\gcd(e, \varphi(N)) = 1$ ed $e \cdot d = 1 \bmod \varphi(N)$.

Esperimento $\text{RSA-inv}_{A, \text{GenRSA}}(n)$

1. Esegui $\text{GenRSA}(1^n)$ per ottenere (N, e, d)
2. Scegli uniformemente $y \in \mathbb{Z}_N^*$
3. A riceve N, e, y e dà in output $x \in \mathbb{Z}_N^*$
4. L'output dell'esperimento è 1 se $x^e = y \bmod N$; 0 altrimenti

Definizione 8.46 Il problema RSA è difficile relativamente a GenRSA se, $\forall A$ PPT, $\exists$ una funzione negl trascurabile, tale che:

$$\Pr\left[\text{RSA-inv}_{A, \text{GenRSA}}(n) = 1\right] \leq \text{negl}(n)$$

L'assunzione RSA $\iff$ $\exists$ GenRSA relativamente al quale il problema RSA è difficile.

Uno scheletro per l'algoritmo GenRSA è:

Algoritmo 8.§7

GenRSA – descrizione ad alto livello

Input: parametro di sicurezza 1^n

Output: N, e, d

$(N, p, q) \leftarrow$ GenModulus (1^n)

$\varphi(N) := (p-1)(q-1)$

scegli $e > 1$ tale che $\gcd(e, \varphi(N)) = 1$

calcola $d = [e^{-1} \bmod \varphi(N)]$

return (N, e, d)

Esempio 8.§8

GenModulus $\longrightarrow (N, p, q) = (143, 11, 13)$.

Risulta $\varphi(N) = 120$. Scegliamo $e = 7$. Usando l'alg. di Euclide esteso, $d = 103$ ($\underbrace{7 \cdot 103}_{721} = 1 \bmod 120$)

Quindi, GenRSA dà in output $(143, 7, 103)$

Sia $y = 66$. Il problema consiste nel calcolare la radice 7-ima di 66 modulo 143 senza conoscere d o la fattorizzazione di N.

Calcolare le radice 7-ime modulo N diventa facile se d, $\varphi(N)$ o la fattorizzazione di N sono noti.

Asimmetria
- Difficile senza d o la fattorizzazione di N
- Facile con d

↓

Base per applicazioni del problema RSA alla crittografia a chiave pubblica.

Esempio 8.69

La radice 7-ima di 66 modulo 143 usando $d = 103$ può essere calcolata come:

$$66^d = 66^{103} \bmod 143 = 25 \bmod 143$$

Si può verificare che: $25^e = 25^7 = 66 \bmod 143$.

Scelta di e. Più popolari

$e = 3$. Calcolare le potenze modulo N richiede due moltiplicazioni. Veloce.

Si scelgono p, q tali che $p, q \neq 1 \bmod 3$ in modo tale che $\gcd(3, \varphi(N)) = 1$

Questa scelta è soggetta ad attacchi che sfruttano appunto esponenti piccoli (low exponents attacks)

$e = 2^{16} + 1 = 65537$ (potenza prima con peso di Hamming basso)

Leggermente più costosa di $e = 3$ ma rimuove i vincoli su p e q.

Osservazione: d "piccolo" è una cattiva idea. (attacchi di forza bruta ...)

anche se $d \approx N^{1/4}$ (attacchi di forza bruta non sono applicabili) esistono algoritmi noti per recuperare d da N ed e.

Che relazione sussiste tra l'assunzione RSA e l'assunzione della Fattorizzazione?

Se N può essere fattorizzato $\Rightarrow$ $\varphi(N)$ può essere calcolato

$\Downarrow$

Pertanto: $\Leftarrow$ d può essere calcolato.

Il problema RSA per essere difficile relativamente a GenRSA

$\Downarrow$

Il problema della Fattorizzazione deve essere difficile relativamente a GenModulus

In altri termini:

Il problema RSA NON PUÒ essere più difficile della fattorizzazione.

La difficoltà della fattorizzazione relativamente a GenModulus può soltanto essere una assunzione più debole della difficoltà del problema RSA.

Cosa sappiamo circa l'altra direzione?

Difficoltà della Fattorizzazione $\overset{?}{\Longrightarrow}$ Difficoltà di RSA

(grossa) questione APERTA

La cosa migliore che si riesce a provare è che

"Calcolare d da (N, e) è così difficile quanto fattorizzare"

(Ovviamente, tale risultato non esclude che il problema RSA possa essere attaccato in altri modi che non richiedono la fattorizzazione di N)

Teorema 8.50. Esiste un algoritmo PPT che, dati in input un intero composto N e gli interi e, d con $e \cdot d = 1 \bmod \varphi(N)$, dà in output un fattore di N, eccetto con probabilità trascurabile in $\|N\|$.

Schemi basati sull'assunzione RSA

Plain RSA (1978, Rivest, Shamir, Adleman)

Algoritmo 11.25

GenRSA – RSA Key generation

Input: 1^n

Output: N, e, d

$(N, p, q) \leftarrow$ GenModulus (1^n);

$\varphi(N) = (p-1)(q-1)$

choose $e > 1$ such that $\gcd(e, \varphi(N)) = 1$

compute $d := [e^{-1} \bmod \varphi(N)]$

return (N, e, d)

Dati N, e, d, la cifratura RSA di m

$c = m^e \bmod N$ poggia sul fatto che chi conosce d

può recuperare m calcolando

$$c^d = (m^e)^d = m^{ed} = m \bmod N.$$

D'altra parte, senza d (anche se N ed e sono noti) l'assunzione RSA implica che è difficile recuperare m da c.

Costruzione 11.26

Sia GenRSA definito come in precedenza

- Gen: su input 1^n esegui $\text{GenRSA}(1^n) \to (N, e, d)$

  $\underbrace{(N, e)}_{PK}$ - chiave pubblica, $\underbrace{(N, d)}_{SK}$ - chiave privata.

- Enc: su input $PK = \langle N, e \rangle$ ed $m \in \mathbb{Z}_N^*$,

  calcola il cifrato

  $$c := [m^e \bmod N]$$

- Dec: su input $SK = \langle N, d \rangle$ e $c \in \mathbb{Z}_N^*$,

  calcola il messaggio

  $$m := [c^d \bmod N]$$

Esempio giocattolo

$(N, e, d) \leftarrow GenRSA()$

$(391, 3, 235)$

↓

$17 \cdot 23 \rightarrow \varphi(391) = 16 \cdot 22 = 352 \rightarrow 3 \cdot 235 = 1 \bmod 352$

$p \quad q$

$PK = \langle 391, 3 \rangle, \qquad SK = \langle 391, 235 \rangle$

Per cifrare il messaggio $m = 158 \in \mathbb{Z}^*_{391}$ con $PK = \langle 391, 3 \rangle$

basta calcolare $c := [158^3 \bmod 391] = 295.$

Per decifrare c, il ricevente calcola

$$[295^{235} \bmod 391] = 158.$$

Circa la sicurezza...

La Fattorizzazione $\Rightarrow$ è computazionalmente inammissibile

(assunzione) calcolare la chiave privata dalla pubblica.

L'assunzione RSA $\Rightarrow$ se il messaggio m è scelto unif. a caso in $\mathbb{Z}_N^*$, da c, N ed e è computazionalmente inammissibile calcolare m.

Tuttavia, sono garanzie deboli rispetto alle nozioni di sicurezza che vogliamo ottenere.

e.g. se il messaggio m non è scelto unif. in $\mathbb{Z}_N^*$?

se Adv è interessato ad informazioni parziali?

Inoltre, RSA-(Plain) è deterministico.

- stesso messaggio / stesso cifrato, non può essere sicuro.

Sono noti diversi attacchi generici e specifici: (vedi libro di Goldreich)

- algoritmi che passano per la fattorizzazione di tempo quasi esponenziale - migliori di quelli immaginati
- attacchi che sfruttano "e" piccolo
- attacchi che sfruttano informazioni parziali su m
- attacchi che sfruttano messaggi related o cifrature multiple di m

Padded RSA

Idea: concatenare al messaggio m un pad casuale prima della cifratura

↓

di lunghezza appropriata.

Costruzione 11.30

Sia GenRSA un generatore di chiavi RSA, e sia ℓ una funzione tale che $\ell(n) \leq 2n - 5$, per tutti gli n.

Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$. Dà in output

$PK = \langle N, e \rangle$ ed $SK = \langle N, d \rangle$

Enc: su input $PK = \langle N, e \rangle$ ed $m \in \{0,1\}^{\|N\| - \ell(n) - 2}$,

sceglie $r \in \{0,1\}^{\ell(n)}$ ed interpreta

$\hat{m} := r \| m$ come elemento di $\mathbb{Z}_N^*$

Calcola $c := [\hat{m}^e \bmod N]$

Dec: su input $SK = \langle N, d \rangle$ e $c \in \mathbb{Z}_N^*$, calcola

$\hat{m} := [c^d \bmod N]$

e dà in output gli $\|N\| - \ell(n) - 2$ bit meno significativi di $\hat{m}$.

Osservazione:

$$\begin{array}{ccc} m & \longrightarrow & \hat{m} \\ \downarrow & & \downarrow \\ \{0,1\}^{\|N\|-\ell(n)-2} & & \mathbb{Z}_N^* \end{array}$$

è un mapping reversibile e randomizzato

Inoltre, permette di vedere i messaggi come stringhe binarie di lunghezza al più $\|N\| - \ell(n) - 2$.

La sicurezza di Padded RSA dipende da ℓ.

Se è troppo piccolo ℓ, possono essere tentati tutti i possibili valori ed applicato l'attacco che sfrutta la conoscenza parziale di $\hat{m}$ per decifrare c, per ognuno di essi.

D'altra parte si può dimostrare che se il PAD è il più grande possibile, i.e., m è un SINGOLO BIT, allora Padded RSA è CPA-sicuro.

Per tutti i casi intermedi, la situazione non è chiara

- non ci sono prove che attacchi PPT non possono esistere se vale l'assunzione RSA, ma al momento non se ne conoscono.

RSA OAEP

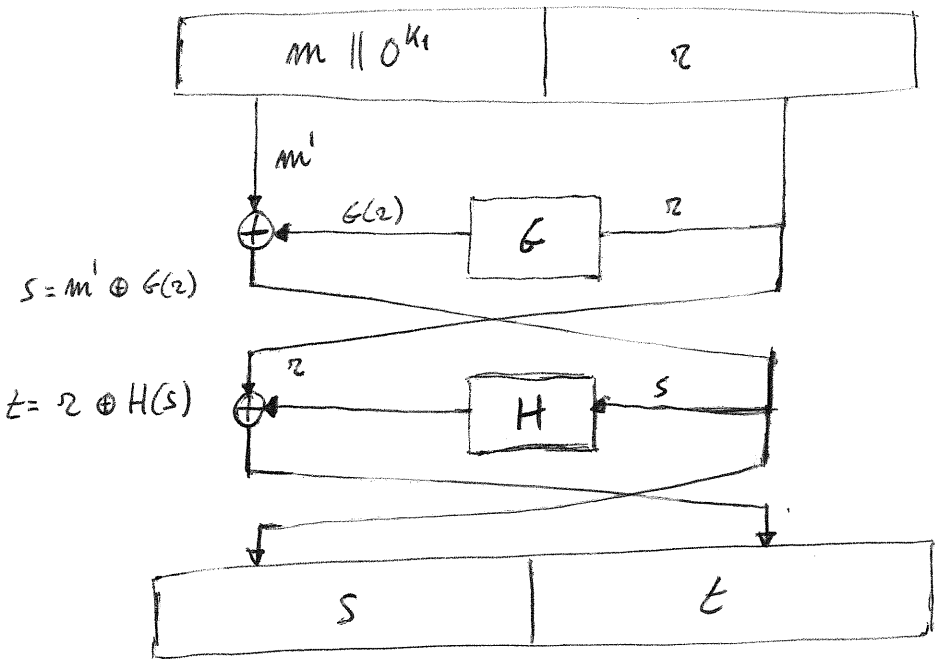
(Optimal asymmetric encryption padding)

Anche in questo caso basata su RSA con padding

$$m \xrightarrow[\text{randomizzata}]{\text{trasformazione}} \hat{m} \in \mathbb{Z}_N^* \longrightarrow c = [\hat{m}^e \bmod N]$$

La trasformazione è però più complessa di prima: usa una rete di Feistel e due funzioni hash.

Una versione di RSA OAEP fa parte dello standard RSA PKCS #1 a partire dalla versione 2.0.



$$\hat{m} = s \,||\, t$$

TRASFORMAZIONE CON RETE DI FEISTEL A DUE ROUND

Costruzione 11.36

Sia GenRSA un generatore di par. RSA ed ℓ, k_0 e k_1 interi.

Sia $G: \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1}$ ed $H: \{0,1\}^{\ell+k_1} \to \{0,1\}^{k_0}$

$\Pi = (Gen, Enc, Dec)$ è definito come segue

- Gen: $(N, e, d) \leftarrow$ GenRSA. Output $PK = \langle N, e \rangle$, $SK = \langle N, d \rangle$.

- Enc: su input $PK = \langle N, e \rangle$ ed $m \in \{0,1\}^{\ell}$, pone $m' := m \| 0^{k_1}$ e sceglie $r \leftarrow \{0,1\}^{k_0}$. Calcola

$$s := m' \oplus G(r), \quad t := r \oplus H(s)$$

pone $\hat{m} := s \| t$. Dà in output $c := [\hat{m}^e \bmod N]$

- Dec: su input $SK = \langle N, d \rangle$ e $c \in \mathbb{Z}_N^*$, calcola $\hat{m} := [c^d \bmod N]$. Se $\|\hat{m}\| > \ell + k_0 + k_1$, $\perp$.

Altrimenti, divide $\hat{m}$ in $s \| t$ con $s \in \{0,1\}^{\ell+k_1}$ e $t \in \{0,1\}^{k_0}$. Calcola

$$r := H(s) \oplus t \quad \text{ed} \quad m' := G(r) \oplus s$$

Se i primi k_1 bit di m' sono $\neq 0$, $\perp$

meno significativi

Altrimenti dà in output gli ℓ bit più significativi di $\hat{m}$.

La dimostrazione è piuttosto complicata ma si può dimostrare che: se il problema RSA è difficile relativamente a GenRSA e le funzioni H e G sono modellate come oracoli casuali, allora RSA-OAEP è CCA-sicuro.

Diamo però un'intuizione del perché.

Durante la cifratura, il mittente calcola:

$$m' := m \| 0^{k_1}, \quad s := m' \oplus G(r), \quad t := r \oplus H(s)$$

($G(r)$: ↓ uniforme)

Il cifrato è $c := [(s \| t)^e \bmod N]$.

Se Adv NON chiede r all'oracolo G, $G(r)$ è uniforme per Adv e quindi m è protetto come nel ONE-TIME PAD.

Potrebbe Adv inviare/chiedere r a G? Nota che il valore r è protetto da $H(s)$. Quindi, se Adv non chiede s all'oracolo H, ancora r è protetto come nel ONE-TIME PAD.

Se la lunghezza di r (k_0) è suff. grande, la prob di

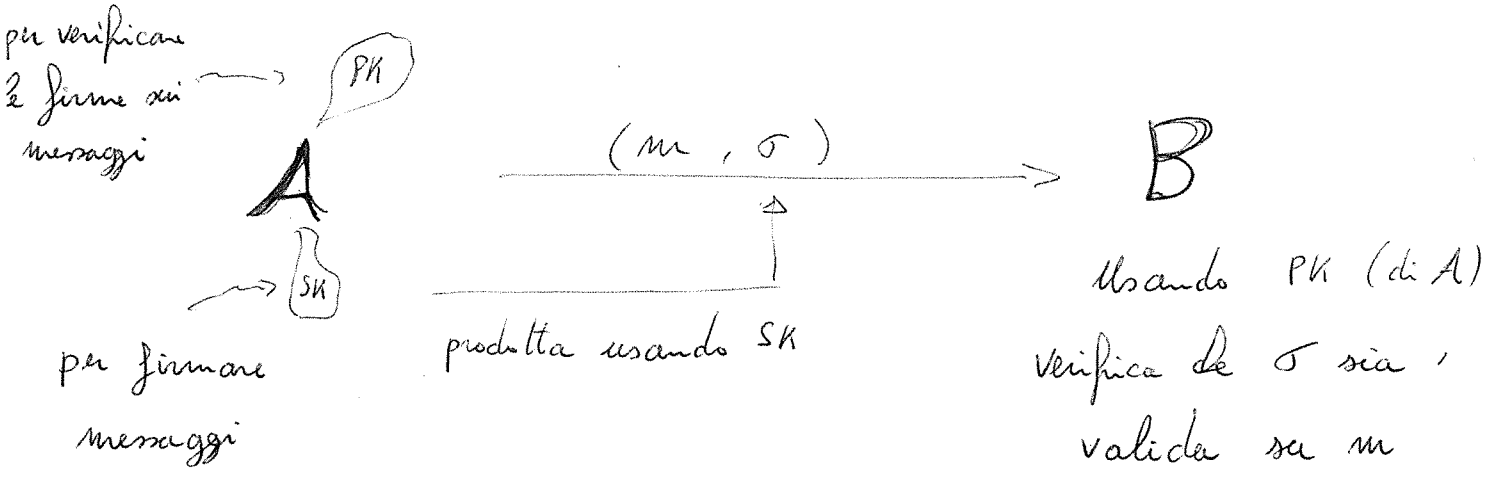
Firme digitali

Schemi di cifratura a chiave pubblica $\Longrightarrow$ Confidenzialità nello scenario a chiave pubblica

Schemi di firma digitale $\Longrightarrow$ Integrità / Autenticità

$\downarrow$

Possono essere visti come l'analogo dei codici per l'autenticazione dei messaggi nel contesto simmetrico

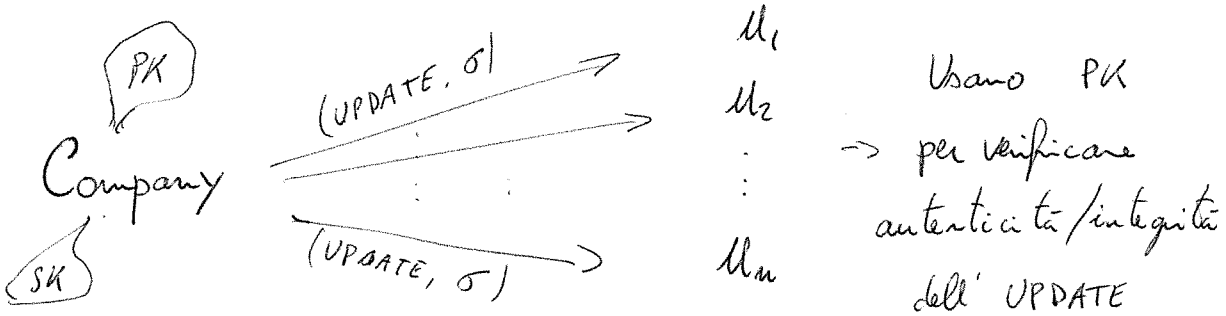


Al contrario, rispetto agli schemi di cifratura, il proprietario della chiave pubblica PK agisce come "mittente".

B, o chiunque altro, può verificare la validità di σ su m usando PK.

Assunzione: le chiavi pubbliche degli utenti sono autenticate. Non ci sono cioè dubbi sull'associazione utente — chiave.

Applicazione tipo: software update



Differenze tra firme e Mac

- Con le firme non occorre più un Mac diverso per ogni ricevente
- <u>Pubblicamente verificabili</u>
- Trasferibilità di una firma.

$$A \xrightarrow{(m,\sigma)} B \xrightarrow{(m,\sigma)} C$$

B trasferisce; C verifica σ su m

- Non ripudio. Una volta che A ha firmato un documento, non può successivamente negare.

- le firme digitali rispetto ai Mac sono più lunghe e 2 o 3 ordini di grandezza più lente da generare / verificare.

Osservazione: le firme digitali vengono spesso viste come l'inverso di uno schema di cifratura a chiave pubblica. Storicamente è stato suggerito di

- firmare, "decifrando" m con SK
- verificare, "cifrando" σ con PK e controllando che coincida con m.

<=> uso delle chiavi "invertito".

Purtroppo questo approccio in generale non ha fondamento. In molti casi è inapplicabile e, se applicabile, le costruzioni ottenute non sono sicure.

Definizioni

Definizione 12.1 Uno schema di firma digitale consiste di tre algoritmi probabilistici di tempo polinomiale $(Gen, Sign, Vrfy)$ tali che:

1. $(pk, sk) \leftarrow Gen(1^n)$ — 1^n: parametro di sicurezza; pk: chiave pubblica; sk: chiave privata

2. $\sigma \leftarrow Sign_{sk}(m)$ — sk: chiave privata; m: messaggio

3. $b := Vrfy_{pk}(m, \sigma)$ — σ: firma; pk: chiave pubblica; m: messaggio; $b = 1$: valida; $b = 0$: invalida

Si richiede che, eccetto con probabilità trascurabile su (pk, sk) (output di $Gen(1^n)$), risulti

$$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$$

per ogni messaggio (legale) m.

Nota: se $M = \{0,1\}^{\ell(n)}$, $(Gen, Sign, Vrfy)$ schema di firma per messaggi di lunghezza $\ell(n)$.

Sicurezza di uno schema di firma digitale

Contraffazione (forgery): coppia (m, σ) dove σ è una firma valida su m non prodotta da S con SK.

Intuizione: un Adv non dovrebbe essere in grado di produrre contraffazioni anche se ottiene le firme di svariati messaggi a sua scelta.

$Sig\text{-}forge_{A,\Pi}(n)$: $\Pi = (Gen, Sign, Vrfy)$

1. $(PK, SK) \leftarrow Gen(1^n)$;
2. $A(PK)$ ha accesso all'oracolo $Sign_{SK}(\cdot)$. Da' in output (m, σ). Sia Q l'insieme delle query che A ha rivolto all'oracolo
3. A ha successo se
   a) $Vrfy_{PK}(m, \sigma) = \underline{1}$ e
   b) $m \notin Q$

In questo caso, l'output dell'exp. è $\underline{1}$. (o altrim.)

Definizione 12.2. Uno schema di firma $\Pi = (Gen, Sign, Vrfy)$ è esistenzialmente infalsificabile (non contraffabile) rispetto ad un attacco adattivo di tipo chosen-message, o semplicemente sicuro, se per ogni A PPT, esiste una funzione trascurabile negl tale che:

$$Pr\left[Sig\text{-}forge_{A,\Pi}(n) = 1\right] \leq negl(n).$$

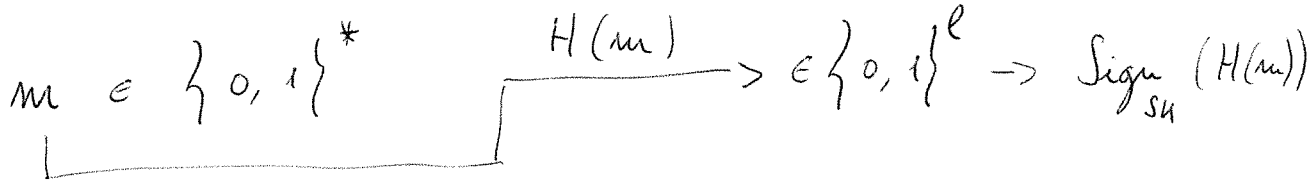
■

Come per gli schemi di cifratura a chiave pubblica, l'approccio ibrido permette di guadagnare efficienza, così per gli schemi di firma un approccio ibrido permette di ottenere schemi più performanti.

Nel caso delle firme questo approccio è costituito dal PARADIGMA HASH-AND-SIGN.

Procede sulla falsariga dell'approccio HASH-AND-MAC.

Disponendo di uno schema di firma per messaggi di lunghezza fissa ℓ, si procede come segue:



Costruzione 12.3

Sia $\Pi = (Gen, Sign, Vrfy)$ per $m \in \{0,1\}^\ell$ e sia $\Pi_H = (Gen_H, H)$ una funzione hash con output di ℓ bit.

$\Pi' = (Gen', Sign', Vrfy')$ è definito come segue

- Gen': su input 1^n, $(PK, SK) \leftarrow Gen(1^n)$;

  $s \leftarrow Gen_H(1^n)$; la chiave pubblica è (PK, s).

  la chiave privata è (SK, s).

- $Sign'$: su input (SK, s) ed $m \in \{0,1\}^*$, dà in output

$$\sigma \leftarrow Sign_{SK}(H^s(m));$$

- $Vrfy'$: su input (PK, s) ed $m \in \{0,1\}^*$ e σ,

  dà in output 1 se e solo se $Vrfy_{PK}(H^s(m), \sigma) = 1$.

Teorema 12.4. Se Π è uno schema di firma digitale per messaggi di lunghezza l sicuro e Π_H è una funzione hash resistente a collisioni, allora la Costruzione 12.3 è uno schema sicuro per messaggi di lunghezza arbitraria.

Schemi di firma: costruzioni

Plain RSA (insicuro, ma esempio per cominciare)

Costruzione 12.5

GenRSA generatore di parametri RSA. $\Pi = (Gen, Sign, Vrfy)$ definito come segue:

- Gen: $(N, e, d) \leftarrow GenRSA(1^n)$.
  chiave pubblica (N, e). chiave privata (N, d).
- $Sign$: su input $sk = (N, d)$ ed $m \in \mathbb{Z}_N^*$,
  calcola $\sigma := [m^d \bmod N]$
- $Vrfy$: su input $pk = (N, e)$, $m \in \mathbb{Z}_N^*$ e $\sigma \in \mathbb{Z}_N^*$,
  da' in output 1 se e solo se $m = [\sigma^e \bmod N]$.

È facile verificare che, per firme generate correttamente, risulta:

$$\sigma^e = (m^d)^e = m^{[ed \bmod \varphi(N)]} = m^1 = m \bmod N$$

Circa la sicurezza, poiché il problema RSA è ritenuto difficile, si potrebbe pensare che sia automaticamente difficile produrre contraffazioni. Si noti che

RSA (assunzione) $\Longrightarrow$ difficile produrre contraffazioni di messaggi m scelti uniformemente

Non assicura nulla sulla difficoltà di produrre firme di messaggi non uniformi (o scelti da Adv).

Un attacco senza messaggi. Adv usa soltanto (N, e).

Data (N, e), scegli $\sigma \in \mathbb{Z}_N^*$, calcola $m := [\sigma^e \bmod N]$ e dà in output (m, σ)!

Ovviamente è una contraffazione. Firma σ valida su m mai generata dal legittimo firmante.

Adv non ha controllo su m, ma è sufficiente a dimostrare che la costruzione non soddisfa la definizione.

Adv, inoltre, potrebbe avere controllo "parziale".

Contraffazione di un msg arbitrario.

Adv riceve due firme dal firmante

Produce una contraffazione su m a sua scelta.

Adv sceglie $m_1, m_2 \in \mathbb{Z}_N^*$ tali che $m = m_1 \cdot m_2 \bmod N$.

Ottiene σ_1 e σ_2 su m_1 ed m_2, rispettivamente.

Dà in output $\sigma := [\sigma_1 \cdot \sigma_2 \bmod N]$ su m.

Risulta:

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = (m_1^{ed} \cdot m_2^{ed}) = m_1 \cdot m_2$$

$$= m \bmod N.$$

L'attacco può essere generalizzato: da un insieme di q firme su $M = \{m_1, \dots m_q\}$, Adv può produrre firme su $2^q - q$ altri messaggi. Devastante!

RSA-FDH e PKCS #1 v2.1

Idea: applicare una trasformazione prima di firmare

$$m \longrightarrow H(m) \longrightarrow \sigma := (H(m))^d \bmod N$$

Costruzione 12.6

GenRSA generatore di parametri RSA

- Gen: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$.

  Chiave pubblica $PK = (N, e)$. Chiave privata $SK = (N, d)$.

  $H: \{0,1\}^* \to \mathbb{Z}_N^*$ specificata

- Sign: su input (N, d) ed $m \in \{0,1\}^*$,

  calcola $\sigma := [H(m)^d \bmod N]$

- Vrfy: su input (N, e), m e una firma σ,

  dà in output 1 se e solo se

  $$\sigma^e \stackrel{?}{=} H(m) \bmod N.$$

Quali proprietà deve avere H?

Sicuramente deve essere difficile da invertire. (primo attacco)

Pensando al secondo attacco, H non deve ammettere "relazioni moltiplicative", e.g. $H(m) = H(m_1) \cdot H(m_2) \bmod N$.

Infine, deve essere difficile trovare collisioni $H(m_1) = H(m_2)$.

Se modelliamo H come un oracolo casuale la costruzione 12.6 è sicura nel Random Oracle Model.

Lo schema è noto come RSA full-domain hash, in breve RSA-FDH.

Intuizione della prova: se $\exists$ A che produce contraffazioni con prob. non trascurabile

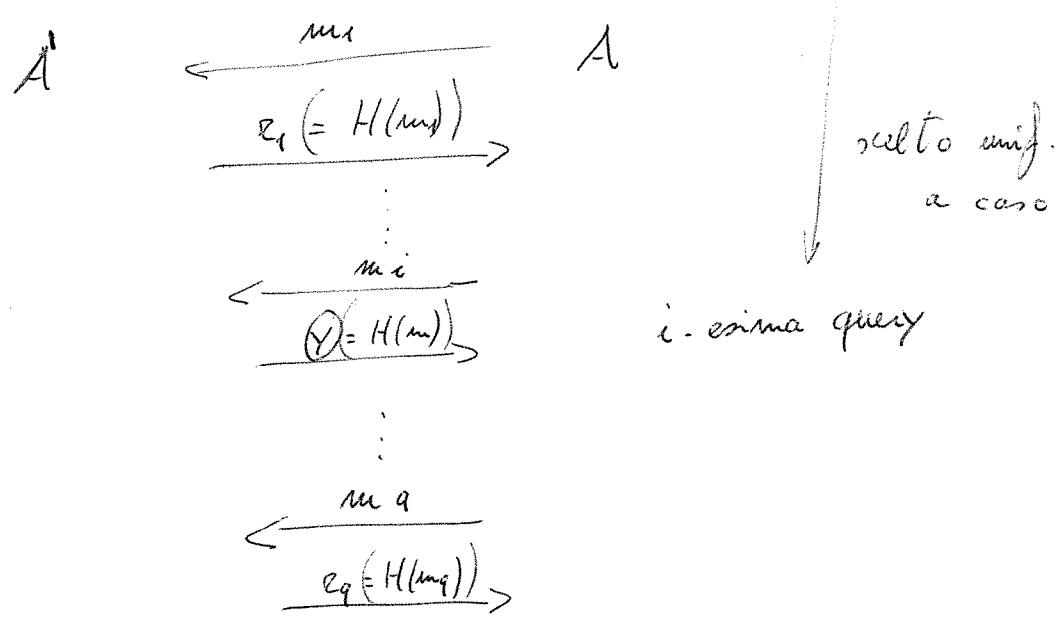
$\Rightarrow$ $\exists$ A' che risolve il problema RSA con prob. altrettanto non trascurabile.

Come lavora A' invocando A? Come segue:

- consideriamo un caso semplice. A non chiede all'oracolo firme su m ma solo valori hash.

A' simula l'esperimento $Sign\text{-}forge_{A,\Pi}(n)$ e risponde alle hash-query di A. Precisamente

A', su input (N, e, y), risponde alle q query di A scegliendo valori casuali in $\mathbb{Z}_N^*$ tranne in un caso. Cioè:



Poiché H è un oracolo casuale i valori $H(m)$ sono uniformi. Pertanto, visto che A' sceglie gli z_j, $j \neq i$, uniformi A non nota nessuna differenza nella distribuzione dei valori che riceve.

Circa y, poiché x è scelto uniformemente, è anch'esso uniforme (RSA è una permutazione).

Quindi, se A produce una contraffazione (m, σ) su un m di cui ha chiesto $H(m)$, con probabilità $\frac{1}{q}$ questo m coincide con m_i!

Pertanto, se A' dà in output σ come soluzione al problema RSA, risulta

$$\sigma^e = H(m) = H(m_i) = y \bmod N$$

con probabilità $\frac{1}{q} \cdot \text{non-negl}(n)$, dove

non-negl(n) è la prob. con cui A genera (m, σ).

A' è efficiente e risolve RSA con prob. non trascurabile.

E se invece A chiede all'oracolo anche firme di messaggi a sua scelta prima di produrre la contraffazione? A' come fa a simulare?

A' non conosce d per firmare.

Idea: A' sceglie, per m_j, un valore

$H(m_j) = \sigma_j^e \bmod N$ per un σ_j scelto a caso e noto quindi.

σ_j uniforme $\Rightarrow$ $H(m_j) = \sigma_j^e$ uniforme come nell'esperimento reale.

Nota: stiamo "programmando" l'oracolo.

Teorema 12.7. Se il problema RSA è difficile relativamente a GenRSA e H è modellata come un oracolo casuale, la costruzione 12.6 è sicura.

RSA PKCS #1 v2.1

Lo standard include uno schema di firma che è una variante di RSA-FDH.

- la firma dipende da un valore casuale (detto SALT) scelto dal firmante all'atto della firma.

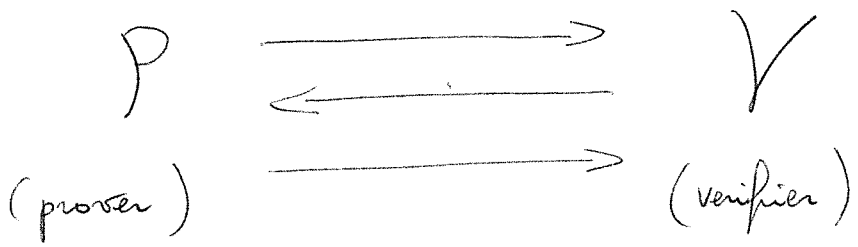
Osservazione: ai fini della sicurezza è importante che il codominio di H sia molto prossimo a tutto $\mathbb{Z}_N^*$. Non è sufficiente usare SHA-1 per esempio. Occorre costruire H attraverso applicazioni ripetute.

Schemi di firma basati sul problema DL

Schema di firma di Schnorr

Per comprenderne la strategia di progettazione discutiamo prima di schemi di identificazione a chiave pubblica

Schema di identificazione: protocollo interattivo che permette ad una parte di provare la propria identità (i.e, autenticare se stesso) ad un'altra parte.

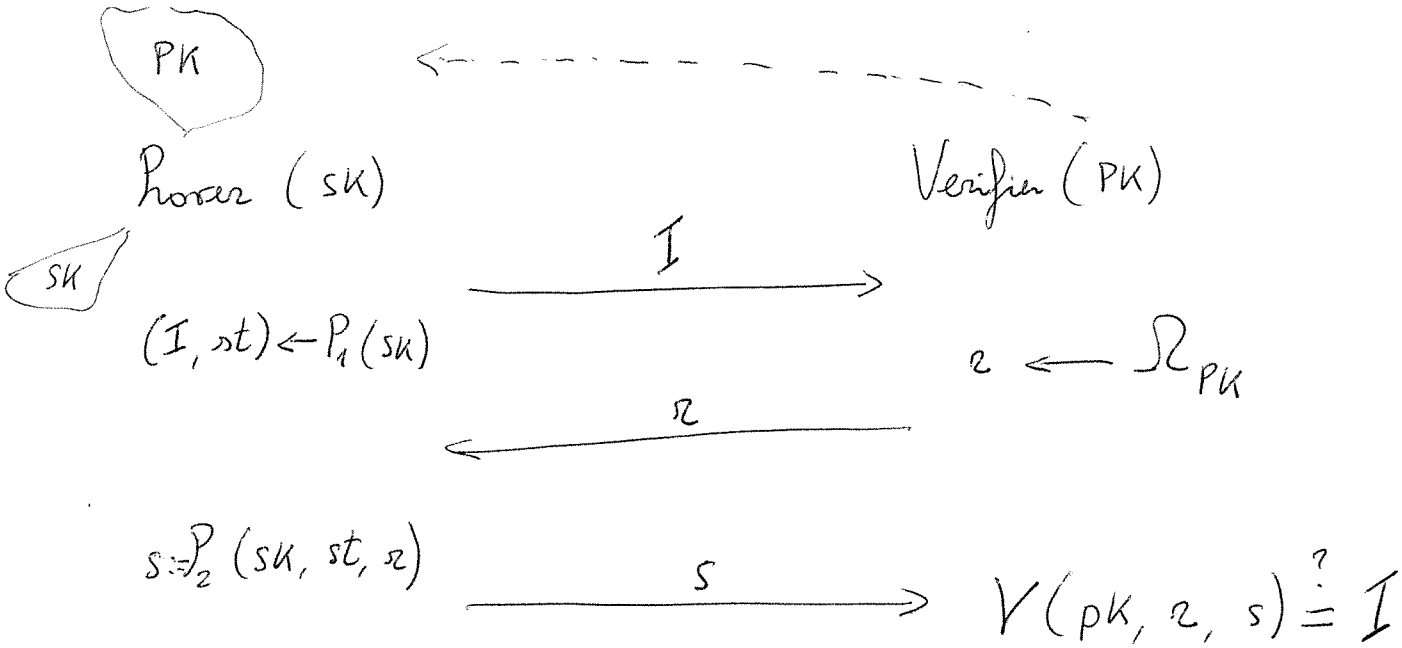


P e V non condividono alcuna informazione a-priori

Consideriamo solo protocolli d'identificazione in tre passi di una forma specifica

P $\xrightarrow{\text{rappresentato da}}$ P_1, P_2 due algoritmi

V $\longrightarrow$ V un solo algoritmo



Si richiede che, se il provatore è legittimo, il verificatore lo identifica sempre.

Consideriamo solo schemi non degeneri: per ogni chiave privata SK, ed ogni messaggio I iniziale, la probabilità che $P_1(SK)$ dia I è trascurabile

Sicurezza?

Ovviamente Adv, senza disporre di SK, non deve essere in grado di farsi identificare da V.

Anche se riesce ad "ascoltare" diverse esecuzioni del protocollo tra P e V.

Formalizzazione. Sia $Trans_{SK}$ un oracolo che, invocato senza input, esegue il protocollo tra due parti e restituisce ad Adv la trascrizione dei messaggi (transcript) che si sono scambiati, ovvero (I, r, s);

Sia $\Pi = (Gen, P_1, P_2, V)$ uno schema di id.

$Ident_{A,\Pi}(n)$

1. $Gen(1^n) \longrightarrow (PK, SK)$
2. $A(PK)$ ha accesso a $Trans_{SK}$. Può interrogarlo quante volte vuole. (# polinomiale)
3. A dà in output I.
   Una challenge uniforme $r \in \Omega_{PK}$ viene scelta e passata ad A. A risponde con s.
   (Può continuare ad inviare query a $Trans_{SK}$ dopo aver ricevuto r).
4. L'esperimento vale 1 se e solo se $V(PK, r, s) \stackrel{?}{=} I$.

Definizione 12.8 Uno schema di identificazione $\Pi = (Gen, P_1, P_2, V)$ è sicuro rispetto ad attacchi passivi, o semplicemente sicuro, se per ogni avversario A PPT, esiste una funzione trascurabile negl tale che

$$Pr[Ident_{A,\Pi}(n) = 1] \leq negl(n).$$

▲

Ovviamente possono anche essere considerati e modellati Adv attivi. Ma non è richiesto per gli schemi di firma digitale.

La trasformazione di Fiat e Shamir offre un metodo per convertire uno schema di identificazione interattivo in uno schema di firma non interattivo.

Idea: il prover esegue il protocollo di identificazione da solo, rimuovendo l'interazione usando una funzione hash.

Costruzione 12.9

Sia (Gen_{id}, P_1, P_2, V) uno schema di identificazione

- Gen: $(PK, SK) \longleftarrow Gen_{id}(1^n)$.

  PK specifica l'insieme delle sfide Ω_{PK}.

  Inoltre si specifica $H: \{0,1\}^* \longrightarrow \Omega_{PK}$.

- Sign: su input SK e un messaggio $m \in \{0,1\}^*$

  1. Calcola $(I, st) \leftarrow P_1(SK)$
  2. Calcola $r := H(I, m)$
  3. Calcola $s := P_2(SK, st, r)$

  e dà in output la firma (r, s).

- Vrfy: su input PK, un messaggio m e la firma (r, s), calcola $I := V(PK, r, s)$ e dà in output 1 se e solo se $H(I, m) \stackrel{?}{=} r$.

Una firma (r, s) è legata ad un messaggio specifico m perché r è funzione sia di I che di m.

Cambiando m $\longrightarrow$ cambia totalmente r

Se H viene modellata come un oracolo casuale che mappa gli input uniformemente su $\mathcal{R}_{PK}$, allora r è unif. distribuito. Discende che per:

Adv è tanto difficile trovare una firma valida (r, s) su un messaggio m quanto lo sarebbe impersonare il prover in una esecuzione onesta del protocollo di identificazione.

Teorema 12.10 Sia Π uno schema di identificazione, e sia Π' uno schema di firme ottenuto applicando la trasformazione di Fiat e Shamir ad esso. Se Π è sicuro ed H è modellato come un oracolo casuale, allora Π' è sicuro.

Lo schema di identificazione di Schnorr è basato sulla difficoltà del problema DL.

Sia $\mathcal{G}(1^n) \longrightarrow (G, q, g)$.

Prover esegue $\mathcal{G}(1^n) \to (G, q, g)$, sceglie $x \in \mathbb{Z}_q$ uniformemente e pone $Y := g^x$.

Chiave pubblica: (G, q, g, Y). Chiave privata: x

Prover (x)		Verifier (G, q, g, Y)
$k \leftarrow \mathbb{Z}_q$		
$I := g^k$	$\xrightarrow{\quad I \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \leftarrow \mathbb{Z}_q$
$s := [e \cdot x + k \bmod q]$	$\xrightarrow{\quad s \quad}$	verifica che $g^s \cdot Y^{-e} = I$

È corretto perché:

$$g^s \cdot Y^{-e} = g^{ex+k} \cdot (g^x)^{-e} = g^k = I$$

Perché risulta sicuro?

Prima osservazione: l'ascolto di esecuzioni non aiuta.

Infatti, Adv può simulare transcript di esecuzioni oneste del protocollo da solo, sfruttando soltanto la chiave pubblica e non conoscendo la chiave privata.

Come? Invertendo l'ordine dei passi.

- prima sceglie indip. ed uniformemente $z, s \in \mathbb{Z}_q$ e pone $I := g^s \cdot y^{-2}$.

Distribuzione reale

(I, z, s)

s → uniforme di $\mathbb{Z}_q$

I → elemento uniforme di G

univocamente determinato come

$s = \log_g (I \cdot y^z)$

uniforme

Distribuzione simulata

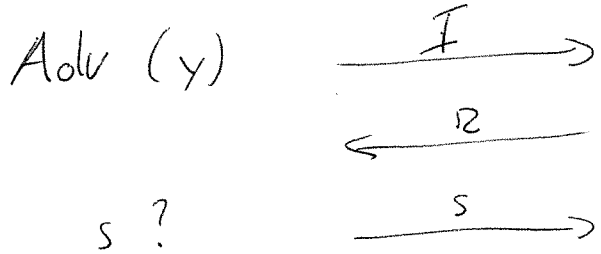
(I, z, s)

z, s <= uniformi in $\mathbb{Z}_q$

I è uniforme in G ed indipendente da z (essendo s uniforme)

Pertanto, le due distribuzioni sono identiche!

Possiamo quindi ridurci all'analisi di Adv che non assoltano.



Se un Adv fosse in grado di calcolare valori di s giusti efficientemente e con alta probabilità, allora sarebbe in grado di calcolare risposte corrette s_1 ed s_2 ad almeno due sfide $r_1, r_2 \in \mathbb{Z}_q$. Nota che:

$$g^{s_1} \cdot y^{-r_1} = I = g^{s_2} \cdot y^{-r_2}$$

da cui $g^{s_1 - s_2} = y^{r_1 - r_2}$. Ciò implica che Adv può implicitamente calcolare

$$\log_g y = [(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q]$$

ovvero il logaritmo discreto di y, contraddicendo la presunta difficoltà del problema DL.

Teorema 12.11. Se il problema del logaritmo discreto è difficile relativamente a $\mathcal{G}$, allora lo schema di identificazione di Schnorr è sicuro. ▲

Lo schema di firma si ottiene applicando Fiat-Shamir.

Costruzione 12.12.

- Gen: $(G, q, g) \leftarrow \mathcal{G}(1^n)$. Sceglie uniformemente $x \in \mathbb{Z}_q$ e pone $y := g^x$. Chiave privata x, pubblica (G, q, g, y). Specificata $H : \{0,1\}^* \to \mathbb{Z}_q$.

- Sign: su input la chiave privata x, ed un messaggio $m \in \{0,1\}^*$, sceglie $K \leftarrow \mathbb{Z}_q$ e pone $I := g^K$. Poi calcola $r := H(I, m)$, seguito da $s := [r \cdot x + K \bmod q]$. Dà in output la firma (r, s).

- Vrfy: su input (G, q, g, y), un messaggio m e una firma (r, s), calcola $I := g^s \cdot y^{-r}$ e dà in output 1 se $H(I, m) \stackrel{?}{=} r$

DSA ed ECDSA

Il Digital Signature Algorithm (DSA) e l'Elliptic Curve Digital Signature Algorithm (ECDSA) sono entrambi basati sul problema DL su classi di gruppi differenti.

Stesso template ⟶ possono essere visti come costruiti da uno schema di identificazione sottostante

Consideriamo il seguente schema di identificazione

Chiave pubblica: (G, q, g, y) con $y = g^x$.

Chiave privata: x.

1. Il provatore sceglie unif. $k \in \mathbb{Z}_q^*$ e manda $I := g^k$.
2. Il verificatore sceglie $a, z \in \mathbb{Z}_q$ uniformi e li invia come challenge
3. Il provatore invia $s := [k^{-1} \cdot (a + xz) \bmod q]$ come risposta
4. Il verificatore accetta se $s \neq 0$ e

$$g^{as^{-1}} \cdot y^{zs^{-1}} \stackrel{?}{=} I.$$

Si noti che $s \neq 0$ a meno di $\alpha = -x r \bmod q$

(che accade con probabilità trascurabile)

$\Rightarrow$ $s^{-1} \bmod q$ esiste e risulta:

$$g^{\alpha s^{-1}} \cdot Y^{r \cdot s^{-1}} = g^{\alpha s^{-1}} \cdot g^{x \cdot r \cdot s^{-1}} = g^{(\alpha + x r) \cdot s^{-1}} = g^{(\alpha + x r) \cdot (\alpha + x r)^{-1} \cdot K}$$

$$= g^{K} = I.$$

Quindi lo schema è corretto. Si può dimostrare inoltre che se il problema DL è difficile relativamente a $\mathcal{G}$.

Sketch della prova:

- transcript di esecuzioni oneste possono essere simulati

  scegli a caso $\alpha, r \in \mathbb{Z}_q$, $s \in \mathbb{Z}_q^*$ e poni $I = g^{\alpha s^{-1}} \cdot Y^{r \cdot s^{-1}}$

  (la simulazione NON è perfetta ma sufficientemente vicina)

- Se Adv dà in output un I per cui può dare risposte corrette $s_1, s_2 \in \mathbb{Z}_q^*$ a challenge distinte (α, r_1) (α, r_2)

  allora $g^{\alpha \cdot s_1^{-1}} \cdot Y^{r_1 s_1^{-1}} = I = g^{\alpha s_2^{-1}} \cdot Y^{r_2 s_2^{-1}} \Rightarrow$

  $g^{\alpha (s_1^{-1} - s_2^{-1})} = Y^{r_2 s_2^{-1} - r_1 \cdot s_1^{-1}} \Rightarrow$ posso calcolare $\log_g Y$.

  Vale anche per challenge (α_1, r_1) (α_2, r_2).

Gli schemi di firma DSA / ECDSA sono costruiti facendo collassare lo schema di identificazione in un algoritmo non interattivo eseguito dal firmante.

Rispetto alla trasformazione di Fiat e Shamir, operiamo come segue:

- Poni $d := H(m)$, m messaggio, H hash function
- Poni $r := F(I)$, $F : G \to \mathbb{Z}_q$, F funzione semplice (no ROM)

In DSA, G sottogruppo di ordine q di $\mathbb{Z}_p^*$ (p primo)

$$F(I) \stackrel{def}{=} [I \bmod q]$$

In ECDSA, G sottogruppo di ordine q di una curva ellittica $E(\mathbb{Z}_p)$, p primo

$$F(\underbrace{(x, y)}_{\text{punto di } E(\mathbb{Z}_p)}) \stackrel{def}{=} [x \bmod q]$$

Entrambi gli schemi possono essere descritti in modo astratto come segue:

Costruzione 12.13

Sia $\mathcal{G}$ un generatore di gruppi

- Gen: su input 1^n, esegui $\mathcal{G}(1^n) \to (\mathbb{G}, q, g)$

  Scegli $x \in \mathbb{Z}_q$ e poni $y := g^x$. La chiave pubblica è $(\mathbb{G}, q, g, y)$ e la chiave privata è x.

  $H : \{0,1\}^* \to \mathbb{Z}_q$, $F : \mathbb{G} \to \mathbb{Z}_q$ fissate

- Sign: su input una chiave privata x ed $m \in \{0,1\}$,

  scegli unif. $k \in \mathbb{Z}_q^*$ e poni $r := F(g^k)$.

  Poi calcola $s := [k^{-1} \cdot (H(m) + xr) \bmod q]$.

  Se $r = 0$ o $s = 0$ ripeti. Dai in output (r, s).

- Vrfy: su input una chiave pubblica $(\mathbb{G}, q, g, x)$,

  un messaggio $m \in \{0,1\}^*$ e una firma (r, s)

  con $r, s \neq 0 \bmod q$, dai in output 1 se e solo se

$$r \stackrel{?}{=} F\left(g^{H(m)\cdot s^{-1}} \cdot y^{r \cdot s^{-1}}\right)$$

Se il problema DL è difficile rel. a $\mathcal{G}$, ed H ed F sono modellati come oracoli casuali, la costruzione è sicura.

Osservazioni. Il valore $K \in \mathbb{Z}_q^*$ deve essere scelto uniformemente a caso. Se una sorgente povera di randomness porta ad un K predicibile, Adv può calcolare da (r, s) su m la chiave privata x!

Infatti: $s = K^{-1}(H(m) + x r) \bmod q$.

$\Rightarrow$ K noto, unica incognita x.

Anche se K è impredicibile, ma usato per produrre DUE firme, la chiave privata x può essere recuperata!

(r, s_1) ed (r, s_2) firme su m_1 ed m_2

$$s_1 = K^{-1}(H(m_1) + x r) \bmod q$$

$$s_2 = K^{-1}(H(m_2) + x r) \bmod q$$

$$\Rightarrow s_1 - s_2 = K^{-1}(H(m_1) - H(m_2)) \bmod q$$

$\Rightarrow$ K può essere calcolato.

$\Rightarrow$ x può essere calcolata procedendo come nel caso precedente.

Applicato per ottenere la master key della Sony PlayStation PS3 nel 2010