signature scheme

- Alice firma un messaggio da mandare a Bob
- ci sono due componenti: un algoritmo sig per firmare e un algoritmo ver per verificare
- quello per firmare dev'essere privato (solo Alice può firmare)
- quello per verificare dev'essere pubblico (Bob - e chiunque altro - può verificare che viene da Alice)
- per firmare il messaggio x Alice usa l'alg sig_k che dipende da una chiave k, e calcola $\text{sig}_k(x) = y$ (lo stesso messaggio può avere diverse firme)
- data una coppia (x, y) dove x è il messaggio e y la firma, l'algoritmo $\text{ver}_k(x, y)$ dà in output vero se y è una firma valida di x, falso altrimenti
- in questo momento, non si chiede che (x, y) sia cifrato

falsificazione

- dato $x \in \mathcal{P}$, dev'essere computazionalmente difficile per chi non è Alice calcolare una firma y tale che $\text{ver}_{k_A}(x, y) = V$
- una coppia (x,y) tale che $\text{ver}_{k_A}(x, y) = V$ che non è stata prodotta da Alice (ma da Eve, per esempio) si dice una falsificazione
- usando lo schema RSA (e anche usando un altro PKCS deterministico)
- è difficile, dato il messaggio x produrre una falsificazione (x, y)
- è però facile data y produrre una falsificazione (x, y)

- Eve può falsificare la firma di Alice
- sceglie y e pone $x = e_{k_A}(y)$
- la coppia (x, y) passa la verifica "per costruzione"
- $\text{ver}_k(x, y) = V \iff x = e_{k_A}(y)$
- una falsificazione di questo tipo si chiama falsificazione esistenziale (existential forgery)
- Eve non può scegliere il messaggio x e produrre una firma valida y (una falsificazione di questo tipo si chiama falsificazione scelta (selective forgery))

funzione hash

- per evitare falsificazioni, si usano schemi di firma insieme a funzioni hash
- molto informalmente, una funzione hash $h : \mathcal{P} \to \mathcal{D}$ è una funzione unidirezionale
- h(x) si dice digest del messaggio x
- può/deve avere molte altre proprietà che non discutiamo

schemi di firma e funzioni hash

- Alice deve firmare il messaggio x
- calcola $h(x)$
- firma il message digest $h(x)$, non x:
  $y = \text{sig}_{k_A}(h(x))$
- Bob riceve (x, y) – per prima cosa, calcola $h(x)$
- poi controlla che $\text{ver}_{k_A}(h(x), y) = V$
- "intuitivamente" questo impedisce le falsificazioni
- Eve sceglie y, calcola $e_{k_A}(y)$ - per produrre una coppia valida, deve trovare $h^{-1}(e_{k_A}(y))$
- lo schema RSA viene sempre usato insieme a una funzione hash

falsificazioni e funzioni hash

- le proprietà delle funzioni hash impediscono le falsificazioni esistenziali
- Eve sceglie y, calcola $e_{k_A}(y)$ - per produrre una coppia valida, deve trovare x tale che $h(x) = e_{k_A}(y)$, quindi $x = h^{-1}(e_{k_A}(y))$
- difficile perché h è preimage reistant
- in un altro attacco Eve ha un messaggio firmato (x, y),
  $y = \text{sig}_{k_A}(h(x))$
- calcola $z = h(x)$, cerca un $\bar{x} \neq x$ con $h(\bar{x}) = h(x)$
- se ci riesce, ha la falsificazione (esistenziale) $(\bar{x}, y)$
- una proprietà della funzioni hash è che deve essere difficile, dato x, trovare $\bar{x} \neq x$ con $h(\bar{x}) = h(x)$
- h deve essere second preimage resistant
- altro possibile attacco: se Eve ha x e $\bar{x}$ tali che $h(\bar{x}) = h(x)$,
- convince Alice a firmare x ottenendo y, e ha la falsificazione $(\bar{x}, y)$
- h deve essere collision resistant

ripetizione: crittosistema Elgamal

- sia p un primo, g un elemento primitivo mod p
- $\mathcal{P} = U(\mathbb{Z}_p)$
- $\mathcal{C} = U(\mathbb{Z}_p) \times U(\mathbb{Z}_p)$
- Lo spazio delle chiavi è

$$\mathcal{K} = \{(p, g, a, \beta) \mid \beta \equiv g^a \pmod{p}\}.$$

- p, g e β sono la chiave pubblica a è la chiave privata

crittosistema Elgamal

- prima di cifrare il messaggio $x \in \mathcal{P}$, Bob sceglie un numero casuale (segreto) $h \in \{2, \ldots, p-2\}$
- $e_k(x, h) = (y_1, y_2)$
- con $y_1 = g^h$, $y_2 = x\beta^h \pmod{p}$
- Alice riceve $(y_1, y_2) \in U(\mathbb{Z}_p) \times U(\mathbb{Z}_p)$ – non conosce h ma conosce a
- calcola $y_1^a = (g^h)^a = (g^a)^h = \beta^h \pmod{p}$
- calcola $(\beta^h)^{-1} \pmod{p}$ e ottiene $x = y_2(\beta^h)^{-1}$
- $d_k((y_1, y_2)) = y_2(y_1^a)^{-1} \pmod{p}$
- notare la somiglianza con DH – non si inverte la funzione $x \to g^x \pmod{p}$
- Bob sceglie un nuovo h a ogni trasmissione

schema di Elgamal

- Alice ha come chiave pubblica Elgamal $k_A = (p, g, \beta)$
  p primo, g elemento primitivo, $\beta = g^a$, a chiave privata
- per firmare $x \in U(\mathbb{Z}_p)$, Alice sceglie h casuale con $(h, p-1) = 1$
- calcola l'inverso l di h modulo $p-1$, quindi $lh \equiv 1 \bmod p-1$
- calcola $z_1 = g^h \bmod p$ e $z_2 = (x - az_1)l \bmod p-1$
- il messaggio firmato è (x, z_1, z_2)
- la verifica funziona – la firma è valida – se $\beta^{z_1} z_1^{z_2} \equiv g^x \bmod p$
- $\beta^{z_1} z_1^{z_2} = g^{az_1} g^{hl(x-az_1)} \bmod p$
- "quindi" $\beta^{z_1} z_1^{z_2} \equiv g^x \bmod p$

esempio

- la chiave pubblica di Alice è $(107, 2, 15)$ e la chiave privata è 11
- quindi $2^{11} \equiv 15 \pmod{107}$
- Alice vuole firmare e trasmettere il messaggio $x = 10$
- sceglie il numero casuale 7
- $(7, 106) = 1$ e $7 \cdot 25 \equiv 1 \pmod{106}$
- la firma di x è (z_1, z_2) con $z_1 = 2^7 \equiv 104 \pmod{107}$
- $z_2 = (10 - 11 \cdot 104) \cdot 25 \equiv 58 \pmod{106}$
- il messaggio firmato è $(10, 104, 58)$
- $\beta^{z_1} z_1^{z_2} = 15^{104} \cdot 104^{58} \equiv 61 \pmod{107}$
- $g^x = 2^{10} \equiv 61 \pmod{107}$
- la verifica ha successo.