Principles of Cryptocurrency Design

Esercitazione

Francesco Pasquale

4 giugno 2026

L'algoritmo di firma digitale ECDSA di un messaggio $\langle msg \rangle$ funziona in questo modo:

Algorithm 1 SIGN($\langle msg \rangle$, `sk`)

1: Calcola $h = \text{HASH}(\langle msg \rangle)$ e assumiamo $h < n$ (se non lo è, si prende il numero formato solo dai $\log n$ bit più significativi);
2: Scegli $k < n$ (possibilmente u.a.r. e con uno pseudorandom generator *sicuro*);
3: Calcola $R = k \cdot G$ e indichiamo con (r, y_r) le sue coordinate;
4: Calcola $s = k^{-1}(h + r \cdot \texttt{sk}) \mod n$
5: Restituisci la firma $\sigma = (r, s)$

Esercizio 1. Verificare che, se conosciamo due messaggi distinti, m_1 e m_2, e due firme, σ_1 e σ_2 di m_1 e m_2 rispettivamente, ottenute usando lo stesso k alla linea `2` dell'Algoritmo 2, allora possiamo calcolare la chiave segreta `sk` con cui sono state generate le firme.

Esercizio 2. Progettare un meccanismo per derivare in modo deterministico e "sicuro" un intero k da usare alla linea `2` dell'Algoritmo 2 a partire dal messaggio $\langle msg \rangle$ da firmare e dalla chiave segreta `sk`. Poi vedere la proposta descritta qui.

Esercizio 3. Leggendo le specifiche sulla nuova serializzazione delle transazioni introdotta con l'upgrade SegWit (si veda, per esempio, qui), modificare opportunamente la classe TRANSACTION, implementata nella lezione del 20 aprile (`https://www.mat.uniroma2.it/~pasquale/dida/aa2526/pcd/pcd260420.zip`), in modo che faccia correttamente anche il *parsing* delle transazioni con la nuova serializzazione.

Siano (1) e (2) rispettivamente le transazioni *Funding* e *Refunding* di un canale *Lightning* fra Alice e Bob

`a5ba...34f9` (*FundingTx*)	
INPUT:	`b8ff...11f0`
OUTPUT:	$10btc$, spendibili da `pk`A & `pk`B

(1)

`e941...05c0` (*RefundingTX*)	
INPUT:	`a5ba...34f9`
OUTPUT:	$10btc$, spendibili da `pk`A, dopo n blocchi oppure da `pk`Ar0 & `pk`B, subito

(2)

Esercizio 4. Usando gli *opcodes* opportuni, progettare un `locking_script` che implementi la condizione descritta nell'output della transazione (2).

Esercizio 5. Spiegare qual è l'ordine con cui devono avvenire i passaggi nell'aggiornamento del canale fra Alice e Bob quando Alice vuole pagare *2btc* a Bob

`5e21...b39d` (*CommitTx1 - Alice*)	
INPUT:	`a5ba...34f9`
OUTPUT 1:	8*btc*, spendibili da `pk`A, dopo n blocchi oppure da `pk`Ar1 & `pk`B, subito
OUTPUT 2:	2*btc*, spendibili da `pk`B

`c401...7a61` (*CommitTx1 - Bob*)	
INPUT:	`a5ba...34f9`
OUTPUT 1:	8*btc*, spendibili da `pk`A
OUTPUT 2:	2*btc*, spendibili da `pk`B, dopo n blocchi oppure `pk`A & `pk`Br1, subito

- L'invio da Alice a Bob della firma di Alice della *CommitTx1 - Bob*;
- L'invio da Bob ad Alice della firma di Bob della *CommitTx1 - Alice*;
- L'invio da Alice a Bob della chiave segreta `sk`Ar1.

Spiegare quale delle due parti potrebbe approfittarne e in che modo se gli scambi di informazione avvenissero in ordine diverso

---

Esercizio 6. Usando gli *opcodes* opportuni, progettare un `locking_script` che implementi la condizione per cui il corrispondente `unlocking_script` sia una firma valida relativa a una chiave pubblica `pk` e la preimmagine (diciamo rispetto a OP_HASH160) un certo h.

Esercizio 7. Usando gli *opcodes* opportuni, progettare un `locking_script` che implementi la condizione per cui il corrispondente `unlocking_script` sia una firma valida relativa a una chiave pubblica `pk` e la preimmagine (diciamo rispetto a OP_HASH160) un certo h.

Il file `https://francescopasquale.it/pcd2425/pcd_lightning_snapshot_250526.zip` contiene uno *snapshot* della rete Lightning, così come era nota al nostro nodo Lightning il 26/05/2025. L'archivio compresso consiste in un fle json con la descrizione dei nodi e degli archi della rete, oltre a tutta una serie di informazioni.

Esercizio 8. Scrivere un programma che legga il file e calcoli il numero totale di nodi, il numero totale di archi e il numero di coppie di nodi fra le quali c'è più di un arco.

Esercizio 9. Scrivere un programma che legga il file, calcoli il numero di componenti connesse del (multi)grafo e il diametro della componente maggiore (quella con il maggior numero di nodi).

L'algoritmo di firma di Schnorr è quello descritto qui di seguito

Algorithm 2 SIGN($\langle msg \rangle$, `sk`)

1: Scegli $k < n$ (possibilmente u.a.r. e con uno pseudorandom generator *sicuro*);
2: Calcola $R = k \cdot G$;
3: Calcola $h = \text{HASH}(\langle msg \rangle \,||\, R)$;
4: Calcola $s = k + h \cdot \texttt{sk}$
5: Restituisci la firma $\sigma = (R, s)$

Si osservi che, come per ECDSA, (1) è cruciale che il valore k scelto alla linea `2` rimanga segreto, altrimenti dal messaggio $\langle msg \rangle$, dalla firma $\sigma = (R, s)$ e da k si può ricavare la chiave segreta `sk` invertendo l'equazione alla linea `4`; (2) è cruciale che non si eseguano mai due firme di due messaggi distinti con lo stesso valore di k.

Esercizio 10. Verificare che, se conosciamo due messaggi distinti, m_1 e m_2, e due firme, σ_1 e σ_2 di m_1 e m_2 rispettivamente, ottenute usando lo stesso k alla linea `2` dell'Algoritmo 2, allora possiamo calcolare la chiave segreta `sk` con cui sono state generate le firme.

L'algoritmo di verifica della firma funziona in questo modo:

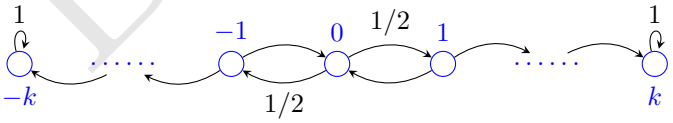
Algorithm 3 VERIFY($\langle msg \rangle$, `pk`, $\sigma = (R, s)$)

1: Calcola $h = \text{HASH}(\langle msg \rangle \,||\, R)$;
2: Return $s \cdot G == R + h \cdot \texttt{pk}$

Esercizio 11. Verificare che lo schema di firma digitale descritto negli Algoritmi 2 e 3 è *corretto*: per ogni coppia di chiavi $(\texttt{sk}, \texttt{pk})$ e per ogni messaggio $\langle msg \rangle$, si ha che

$$\text{VERIFY}\big(\langle msg \rangle, \texttt{pk}, \text{SIGN}(\langle msg \rangle, \texttt{sk})\big) = \text{TRUE}$$

Sia $\{X_t : t \in \mathbb{N}\}$ la seguente una catena di nascita e morte.



Esercizio 12. Sia $\tau = \inf\{t \in \mathbb{N} : X_t = \pm k\}$ la variabile aleatoria che indica il primo istante in cui la catena si trova nello stato k o nello stato $-k$. Dimostrare che il valore atteso di τ per la catena che parte da uno stato arbitrario $j \in \Omega$ è

$$\mathbf{E}\left[\tau \mid X_0 = j\right] = k^2 - j^2$$

Esercizio 13. Sia $\{X_t : t \in \mathbb{N}\}$ una catena di nascita e morte *unbiased* con spazio degli stati $\Omega = \mathbb{Z}$. Per ogni istante $t \in \mathbb{N}$ e per ogni *target state* $k \in \mathbb{N}$ si ha che

$$\mathbf{P}\left(|X_t| \geqslant k \mid X_0 = 0\right) \leqslant \mathbf{P}\left(\max_{i=1,\ldots,t} |X_i| \geqslant k \mid X_0 = 0\right) \leqslant 2\mathbf{P}\left(|X_t| \geqslant k \mid X_0 = 0\right)$$