

Principles of Cryptocurrency Design

Esercitazione

Francesco Pasquale

9 giugno 2025

Sia $(\mathbb{F}_p, +, \cdot)$, con p primo, il campo dei numeri $\{0, 1, \dots, p-1\}$ con le usuali operazioni di somma e prodotto modulo p . Sia $\mathcal{C} = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b\} \cup \{\infty\}$ una curva ellittica. Sia $G \in \mathcal{C}$ un punto della curva tale che l'ordine n del gruppo $\langle G \rangle$ generato da G sia primo. Sia $(\mathbf{sk}, \mathbf{pk})$, con $\mathbf{sk} \in \{1, \dots, n-1\}$ e $\mathbf{pk} = \mathbf{sk} \cdot G$, una coppia di chiavi. Sia HASH una funzione hash crittografica.

Algorithm 1 $\text{SIGN}(\langle msg \rangle, \mathbf{sk})$

- 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 $s = k^{-1}(h + r \cdot \mathbf{sk}) \pmod n$
 - 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 1, allora possiamo calcolare la chiave segreta $\mathbf{sk}$ 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 1 a partire dal messaggio $\langle msg \rangle$ da firmare e dalla chiave segreta $\mathbf{sk}$. 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 14 aprile (<https://www.mat.uniroma2.it/~pasquale/dida/aa2425/pcd/pcd250414.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 (<i>FundingTx</i>)	
INPUT:	b8ff...11f0
OUTPUT:	10btc, spendibili da pkA & pkB

(1)

e941...05c0 (<i>RefundingTX</i>)	
INPUT:	a5ba...34f9
OUTPUT:	10btc, spendibili da pkA, dopo n blocchi oppure da pkAr0 & pkB, 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 (<i>CommitTx1 - Alice</i>)		c401...7a61 (<i>CommitTx1 - Bob</i>)	
INPUT:	a5ba...34f9	INPUT:	a5ba...34f9
OUTPUT 1:	8btc, spendibili da pkA, dopo n blocchi oppure da pkAr1 & pkB, subito	OUTPUT 1:	8btc, spendibili da pkA
OUTPUT 2:	2btc, spendibili da pkB	OUTPUT 2:	2btc, spendibili da pkB, dopo n blocchi oppure pkA & pkBr1, 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 skAr1.

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 [HASH160](#)) un certo h .

Si consideri il meccanismo con cui Alice può inviare un pagamento a Carol instradandolo sul cammino Alice – Bob – Carol:

1. Carol genera un valore qualunque r , ne calcola l'hash $h = H(r)$ e dà h ad Alice;
2. Alice costruisce una transazione con un output che può essere speso da
 - Bob, se Bob esibisce una preimmagine di h ;
 - Alice, se la Blockchain ha almeno x blocchi, dove x è un numero sufficientemente maggiore del valore corrente del numero di blocchi della Blockchain.
3. Bob a sua volta costruisce una trasazione con un output che può essere speso da
 - Carol, se Carol esibisce una preimmagine di h ;
 - Bob, se la Blockchain ha almeno $x - \Delta$ blocchi, dove Δ è un numero piccolo (e.g., 6).

Esercizio 7. Riflettere sul perché se il *locktime* di Alice è x , quello di Bob deve essere di qualche unità inferiore $x - \Delta$.

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 file 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).

DRAFT