Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

5 maggio 2025

Nella lezione di oggi abbiamo implementato la classe `Address` e l'abbiamo usata per generare indirizzi Bitcoin a partire da numeri a 256 bit. Gli esercizi di questa nota si riferiscono al codice scritto in aula, che potete scaricare qui: `https://www.mat.uniroma2.it/~pasquale/dida/aa2425/pcd/pcd250505.zip`

Esercizio 1. A lezione abbiamo generato un indirizzo bitcoin-testnet a partire dal numero

$$r = \texttt{1032026271963554642597181681368518600454288997291561949319905424087874760107 2}$$

e abbiamo inviato a quell'indirizzo 0.001 bitcoin-testnet. Tovare il modo di spostare quei 0.001 bitcoin-testnet inviandoli a un altro indirizzo sotto il vostro controllo.

Esercizio 2. Il Wallet Import Format (WIF) è un formato standard con cui è possibile esportare e importare chiavi private fra wallet bitcoin. Per ottenerlo, dalla sequenza di byte `sk` che costituiscono la chiave privata si procede i questo modo

1. Si aggiunge come prefisso il byte `0xEF` per la *testnet* e il byte `0x80` per la *mainnet*;
2. Se si vuole considerare una public key in formato compresso si aggiunge come suffisso il byte `0x01`
3. Della sequenza di byte così ottenuta si calcola l'`hash256` e i primi quattro byte di questo hash vengono aggiunti come suffisso;
4. Si codifica la sequenza di byte così ottenuta in `base58`

Scrivere un metodo `wif` che restituisca la chiave privata in quel formato. Testare il vostro metodo, generando chiavi private e provando a importarle in un wallet standard, per esempio Electrum.

Esercizio 3. Scrivere un metodo di classe PARSE_WIF che legga una stringa di byte in formato WIF e restituisca l'oggetto corrispondente della classe ADDRESS.

Esercizio 4. La libreria *random* che abbiamo usato a lezione per fare i nostri test non è sicura per generare numeri casuali da usare come chiavi private. Il seguente codice genera una sequenza di 32 byte (256 bit) in modo più sicuro

```
import secrets
sk_string = secrets.token_bytes(32)
print(sk_string.hex())
```

Scrivere un programma che trovi una chiave privata `sk` "sicura" e tale che l'indirizzo Bitcoin ottenuto a partire da quella chiave inizi con `1PCD`.

Esercizio 5. Verificare che uno degli indirizzi contenuti negli output script P2PKH della transazione b35d91a71f226ba961162ca18f321b4d9aada8a0e722430ad3d2d2e4dda9a2c0,
l'indirizzo `1CLJKodMLHhAwiDYFWDiwmz31cMfhqKnEc`, è l'hash di una chiave pubblica non compressa che ha come come chiave privata lo `sha256` della stringa `Francesco`. Quella transazione ha 500 output del tipo P2PKH tutti con lo stesso ammontare. Scrivere un programma *brute-force* che cerchi di individuare se in quella transazione ci sono altri indirizzi che hanno come chiave privata l'hash `sha256` di altri nomi.

Esercizio 6. Ottenere dei bitcoin testnet tramite qualche faucet, provare a eseguire delle transazioni verso indirizzi con chiavi private facilmente individuabili tramite brute-force e vedere quanto tempo "resistono" prima che qualche bot ne individui la chiave privata e li spenda.