# Principles of Cryptocurrency Design

Appunti ed Esercizi

Francesco Pasquale

17 aprile 2025

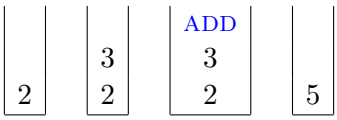
Abbiamo detto precedentemente che una transazione $\texttt{tx}_a$ può avere uno o più output $\texttt{tx}_a\texttt{.output}$, ogni output è costituito da un `ammontare` e da un `locking_script`. Una transazione $\texttt{tx}_b$ che volesse *spendere* $\texttt{tx}_a\texttt{.output}$ deve contenere, nell'input che punta a $\texttt{tx}_a\texttt{.output}$, un `unlocking_script` tale che la concatenazione di $\texttt{tx}_b\texttt{.input.unlocking\_script}$ con $\texttt{tx}_a\texttt{.output.locking\_script}$ formi uno *script* che restituisca TRUE. Vediamo più in dettaglio questo processo.

## 1 Script

Il software del sistema Bitcoin, in esecuzione su ogni nodo della rete, contiene l'iterprete di un semplice linguaggio di programmazione che viene chiamato Script. Il linguaggio Script è *stateless* (non ci sono variabili che possono salvare localmente un dato e andare a rileggerlo successivamente). Un programma (uno *script*) è una sequenza di comandi (*istruzioni* o *dati*) che vengono eseguiti secondo un modello *stack-based*: i *dati* e le *istruzioni* vengono inseriti in una *pila* (lo *stack*); quando una istruzione viene inserita nella pila, può estrarre uno o più dati dalla pila, eseguire delle computazioni sui dati ed eventualmente inserire il risultato delle computazioni nella pila. Facciamo un esempio. L'istruzione ADD nel linguaggio Script estrae due dati dalla pila e ne esegue la somma, quindi il seguente *script*

```
<2> <3> ADD
```

verrebbe eseguito dall'interprete Script presente sui nodi della rete Bitcoin in questo modo: viene inserito nella pila il dato 2, poi viene inserito nella pila il dato 3, poi viene inserito nella pila l'istruzione ADD che estrae dalla pila i due dati sottostanti e inserisce nella pila 5




Il più semplice `locking_script` usato in Bitcoin è chiamato *Pay-to-Public-Key (p2pk)* ed è formato da due comandi:

```
<pk> CHECKSIG
```
(1)

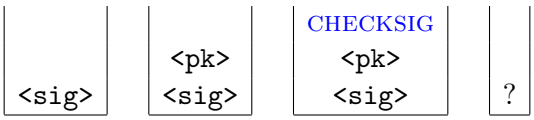
Il primo comando, `<pk>`, è un dato costituito da una chiave pubblica. Il secondo, CHECKSIG, è una istruzione (OP_CODE nella terminologia di Bitcoin) che estrae dalla pila i due comandi sottostanti e verifica se il secondo è una firma corretta della transazione rispetto alla chiave pubblica contenuta nel primo dato, in caso affermativo inserisce nella pila TRUE e la transazione risulta valida, in caso negativo inserisce nella pila FALSE e la transazione viene rigettata come non valida. Se una transazione $\texttt{tx}_a$ ha un output $\texttt{tx}_a\texttt{.output}$ con un `locking_script` di questo tipo, allora per spendere quell'output una transazione $\texttt{tx}_b$ dovrà avere un input con un `unlocking_script` costituito da un solo comando:

```
<sig>
```

in modo che la concatenazione di $\texttt{tx}_b$.input.unlocking_script con $\texttt{tx}_a$.output.locking_script sia

<sig> <pk> CHECKSIG (2)

e l'interprete Script che esegue lo script avrà sullo stack alla fine dell'esecuzione TRUE o FALSE a seconda che <sig> sia oppure no una firma valida per la chiave pubblica <pk> della transazione $\texttt{tx}_b$




Si noti che <sig> non può contemporaneamente essere parte della transazione $\texttt{tx}_b$ ed essere una firma valida per $\texttt{tx}_b$. Infatti, più precisamente possiamo dire che, affinché lo script in (2) inserito in una transazione $\texttt{tx}_b$ ritorni TRUE, <sig> deve essere una firma valida della sequenza di byte che si ottiene da $\texttt{tx}_b$ dove nel campo di $\texttt{tx}_b$.input.unlocking_script riservato a <sig> viene inserito <pk>.

Tutte le prime transazioni di Bitcoin avevano dei locking_script del tipo *p2pk*. Successivamente, diversi nuovi tipi di locking script sono stati introdotti: il secondo, dopo p2pk, è stato quello chiamato *Pay-to-Public-Key-Hash (p2pkh)*. In questo tipo di locking_script non compare direttamente la chiave pubblica, ma l'hash della chiave pubblica <pkhash>. Complessivamente locking_script è formato da cinque comandi:

DUP HASH160 <pkhash> EQUALVERIFY CHECKSIG (3)

dove

- DUP è un'istruzione che inserisce nello stack una copia dell'elemento sottostante. Per esempio, lo script <1> DUP verrebbe eseguito in questo modo


- HASH160 è un'istruzione che estrae dallo stack l'elemento sottostante e inserisce nello stack il suo hash (opportunamente calcolato);
- <pkhash>è l'hash di una chiave pubblica;
- EQUALVERIFY è un'istruzione che estrae due elementi sottostanti dallo stack e se sono uguali non fa nulla, altrimenti interrompe l'esecuzione dello script dichiarando la transazione non valida;
- CHECKSIG è il comando che abbiamo visto in precedenza in *p2pk*.

**Esercizio 1.** Qual è lo unlocking_script che consente di sbloccare un locking_script del tipo *p2pkh*?

**Esercizio 2.** L'istruzione EQUAL estrae due elementi sottostanti dallo stack e inserisce nello stack TRUE se i due elementi sono uguali e FALSE altrimenti. Scrivere un locking_script che può essere sbloccato solo da chi conosce un messaggio msg il cui hash è <msg_hash>.

**Esercizio 3.** L'istruzione NOT estrae un elemento dallo stack e, se è TRUE o FALSE ne inserisce la negazione nello stack, altrimenti inserisce nello stack FALSE. L'istruzione VERIFY estrae un elemento dallo stack e, se è TRUE non fa nulla altrimenti interrompe l'esecuzione e la transazione è non valida. L'istruzione SWAP estrae due elementi dallo stack e li reinserisce nello stack in ordine inverso. Usando queste e le altre istruzioni viste in precedenza, progettare un locking_script che può essere sbloccato solo con un unlocking_script che contenga un messaggio <msg2> diverso da un dato messaggio <msg> che però abbia lo stesso valore di hash (calcolato con HASH160).