UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

ciie
Centro de Investigación
e Innovación Educativa

¿Cómo funciona Bitcoin?

3.1 Transacciones Bitcoin Script

Transacción Bitcoin: A un bajo nivel


```
    ],
    "lock_time":0,
    "size":404,
    "double_spend":false,
    "block_index":345993,
    "time":1391279401,
    "tx_index":49679742,
    "vin_sz":2,
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "vout_sz":1
}
```


Metadato


```
   "ver":1,
   "inputs":[
      {
         "sequence":4294967295,
         "witness":"",
         "prev_out":{
            "spent":true,
            "spending_outpoints":[
               {
                  "tx_index":49679742,
                  "n":0
               }
            ],
            "tx_index":46754344,
            "type":0,
            "addr":"168UB7t2V3GNFBShzei7cEV5zt19V3s2Rh",
            "value":12297097,
            "n":0,
            "script":"76a914384239890ab61e2741c1999927b0bffd24fdaa8388ac"
         },

"script":"47304402204a01ee13feff5146743e9007c46ec2650ef82bd54c8f27c48d2ac7a07320380a022
03d0c311aa377a60b1fc4b8a87253a0df0c87f0bfa4a2ddce27dc21e8b637450f014104e5ea7670c1350ff6
98de4a5c23be93c16b4fbe857c353e66f26eeade6dbb2335ac46062908aaf2ca97656dae08179254c03ceef
1358bbec41fe40d23f3a4ce81"
```


Entradas

Salidas


```
"out":[
   {
      "spent":true,
      "spending_outpoints":[
         {
            "tx_index":49679920,
            "n":0
         }
      ],
      "tx_index":49679742,
      "type":0,
      "addr":"1AepWSsqsrrPJY78ZdtcgoBGQhKJfcydVe",
      "value":1012287097,
      "n":0,
      "script":"76a91469e02e18b5705a05dd6b28ed517716c894b3d42e88ac"
```

Metadata de la Transacción Bitcoin


```
    ],
    "lock_time":0,
    "size":404,
    "double_spend":false,
    "block_index":345993,
    "time":1391279401,
    "tx_index":49679742,
    "vin_sz":2,
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "vout_sz":1
}
```


La metadata incluye el tamaño de la transacción, el número de entradas y el número de salidas. También el hash de toda la transacción que sirve como un ID único para cada transacción. Finalmente, hay un campo "lock_time", que nos sirve para incluir la dimensión del tiempo a nuestras transacciones.

Inputs de la Transacción Bitcoin


```
"ver":1,
"inputs":[
   {
      "sequence":4294967295,
      "witness":"",
      "prev_out":{
         "spent":true,
         "spending_outpoints":[
            {
               "tx_index":49679742,
               "n":0
            }
         ],
         "tx_index":46754344,
         "type":0,
         "addr":"168UB7t2V3GNFBShzei7cEV5zt19V3s2Rh",
         "value":12297097,
         "n":0,
         "script":"76a914384239890ab61e2741c1999927b0bffd24fdaa8388ac"
      },
```

```
"script":"47304402204a01ee13feff5146743e9007c46ec2650ef82bd54c8f27c48d2ac7a07320380a02203d0c311aa377a60b1fc4b8a87253a0df0c87f0bfa4a2ddce27dc21e8
b637450f014104e5ea7670c1350ff698de4a5c23be93c16b4fbe857c353e66f26eeade6dbb2335ac46062908aaf2ca97656dae08179254c03ceef1358bbec41fe40d23f3a4ce81"
```


Las entradas de transacción forman una matriz. Una entrada especifica una transacción anterior, por lo que contiene un hash de esa transacción, que actúa como un puntero de hash. La entrada también contiene el índice de la salida de la transacción anterior que se está reclamando. Y luego hay una firma.

Outputs de la Transacción Bitcoin


```
"out":[
  {
    "spent":true,
    "spending_outpoints":[
      {
        "tx_index":49679920,
        "n":0
      }
    ],
    "tx_index":49679742,
    "type":0,
    "addr":"1AepWSsqsrrPJY78ZdtcgoBGQhKJfcydVe",
    "value":1012287097,
    "n":0,
    "script":"76a91469e02e18b5705a05dd6b28ed517716c894b3d42e88ac"
```


Las salidas están ordenadas como una matriz. La suma de todos los valores de salida debe ser menor o igual a la suma de todos los valores de entrada. Si la suma de los valores de salida es menor que la suma de los valores de entrada, la diferencia es una tarifa de transacción.

Se supone que cada salida va a una clave pública específica, pero en verdad van a un conjunto de comandos. Este campo es un script.

Lenguaje Bitcoin Script o simplemente Script

Metas de diseño:

- Creado solo para Bitcoin (inspirado en Forth)
- Simple
- Compacto
- Basado en Stack (pilas)
- Tiene límites en el tiempo usado y memoria usada
- No admite loops o lazos

Instrucciones de Bitcoin Script

256 Op_codes en total (15 están deshabilitados, 75 reservados):

- Constantes
- Control de flujo
- Pila
- Lógica
- Aritmética
- Cripto
- Locktime

Bitcoin Script: Cripto

Palabra	Opcode	Maleficio	Entrada	Salida	Descripción
OP_RIPEMD160	166	0xa6	en	picadillo	La entrada se realiza mediante RIPEMD-160.
OP_SHA1	167	0xa7	en	picadillo	La entrada es hash utilizando SHA-1.
OP_SHA256	168	0xa8	en	picadillo	La entrada se procesa utilizando SHA-256.
OP_HASH160	169	0xa9	en	picadillo	La entrada se procesa dos veces: primero con SHA-256 y luego con RIPEMD-160.
OP_HASH256	170	0xaa	en	picadillo	La entrada se procesa dos veces con SHA-256.
OP_CODESEPARATOR	171	0xab	Nada	Nada	Todas las palabras de comprobación de firmas solo harán coincidir las firmas con los datos después del OP_CODESEPARATOR ejecutado más recientemente.
OP_CHECKSIG	172	0xac	decir pubkey	Verdadero Falso	Las salidas, entradas y secuencias de comandos de toda la transacción (desde el OP_CODESEPARATOR ejecutado más recientemente hasta el final) están en hash. La firma utilizada por OP_CHECKSIG debe ser una firma válida para este hash y clave pública. Si lo es, se devuelve 1, 0 en caso contrario.
OP_CHECKSIGVERIFY	173	0xad	decir pubkey	Nada / *falla*	Igual que OP_CHECKSIG, pero OP_VERIFY se ejecuta después.
OP_CHECKMULTISIG	174	0xae	x sig1 sig2 ... <número de firmas> pub1 pub2 <número de claves públicas>	Verdadero Falso	Compara la primera firma con cada clave pública hasta que encuentra una coincidencia ECDSA. A partir de la siguiente clave pública, compara la segunda firma con cada clave pública restante hasta que encuentra una coincidencia ECDSA. El proceso se repite hasta que todas las firmas se hayan verificado o no queden suficientes claves públicas para producir un resultado exitoso. Todas las firmas deben coincidir con una clave pública. Debido a que las claves públicas no se vuelven a verificar si fallan en la comparación de firmas, las firmas deben colocarse en el scriptSig usando el mismo orden en que se colocaron sus claves públicas correspondientes en el scriptPubKey o redeemScript. Si todas las firmas son válidas, se devuelve 1, 0 en caso contrario. Debido a un error, un valor no utilizado extra se elimina de la pila.
OP_CHECKMULTISIGVERIFY	175	0xaf	x sig1 sig2 ... <número de firmas> pub1 pub2 ... <número de claves públicas>	Nada / *falla*	Igual que OP_CHECKMULTISIG, pero OP_VERIFY se ejecuta después.

Bitcoin Script: Ejecución de una transacción

Apilar	Guión	Descripción
Vacío.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Se combinan scriptSig y scriptPubKey.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Las constantes se agregan a la pila.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	El elemento de la pila superior está duplicado.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	El elemento de la pila superior está hash.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constante añadida
<sig> <pubKey>	OP_CHECKSIG	La igualdad se verifica entre los dos elementos de la pila superior.
cierto	Vacío.	La firma se comprueba para los dos elementos de la pila superior.

Bitcoin Script: OP_CHECKMULTISIG

Esta operación está diseñada para:

- Soportar firmas en conjunto (2 o más participantes)
- Especificar “n” claves públicas
- Especificar “t”
- La verificación requiere “t” de las “n” firmas.

Nota: Alerta de Bug: Hay un valor extra que emerge del stack debe ser ignorado.

Bitcoin Script: OP_RETURN

Es un script para marcar la transacción como inválida, dado que los outputs de la transacción no se pueden gastar, esto se utiliza para “quemar monedas”.

Este script también se utiliza para almacenar información arbitraria en la blockchain. Esto se lo llama prueba de pertenencia, porque nos permite demostrar que éramos propietarios de esa información en un momento dado del tiempo.