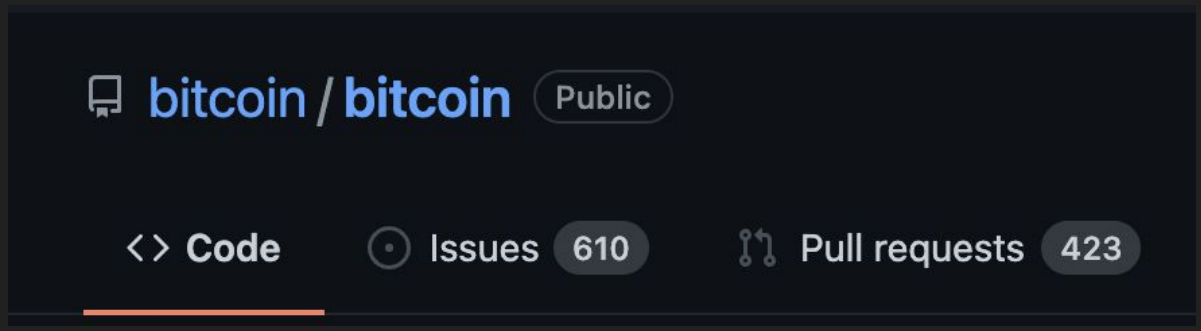
Platzi

Curso de Introducción a Bitcoin Core y Script

Módulo 1:
Bitcoin Core

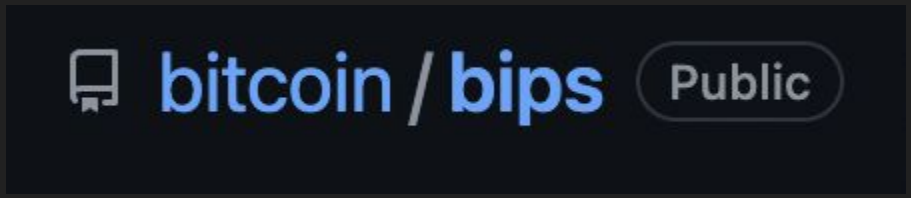
¿ Qué es Bitcoin Core ?

- Implementación de referencia.
- Open source (Licencia MIT).
- Software más usado por la comunidad.
- Referencia para otras implementaciones (líder).
- Bitcoin core es el libro de reglas de Bitcoin.

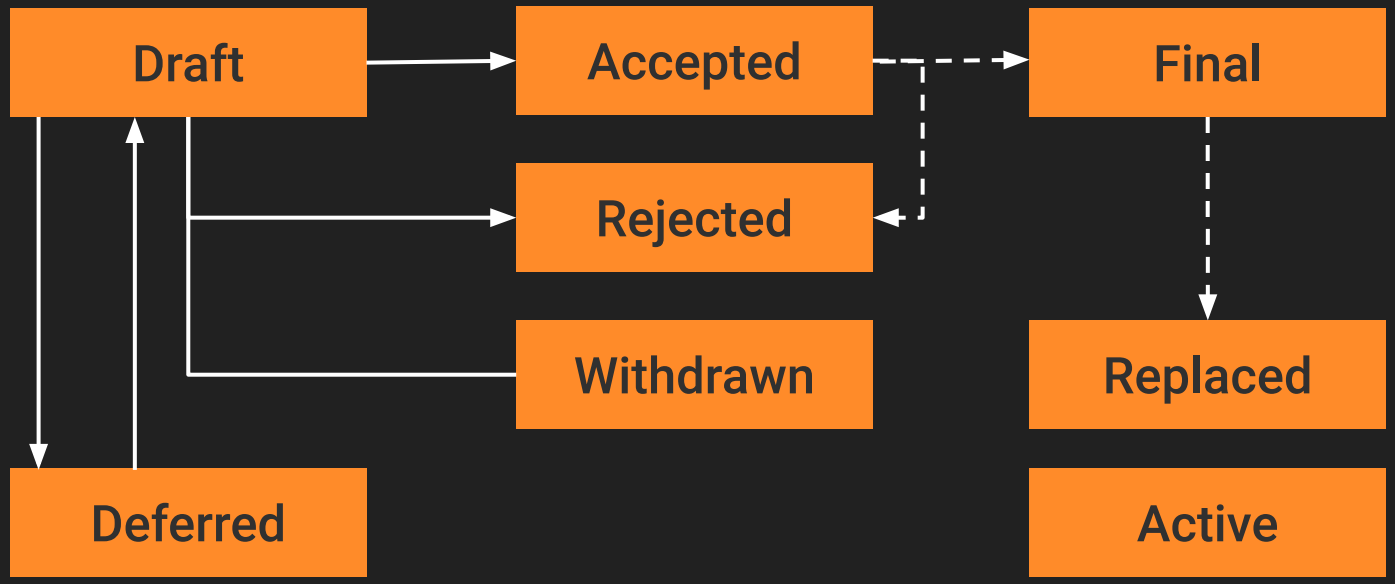


BIP

- Propuestas para cambios en Bitcoin.
- Se publica en serie.
- Cada BIP tiene un líder para evangelizar y coordinar.
- BIPs existentes para informar y orientar procesos en la comunidad.



Flujo BIP


Draft
Accepted
Final
Rejected
Withdrawn
Replaced
Deferred
Active

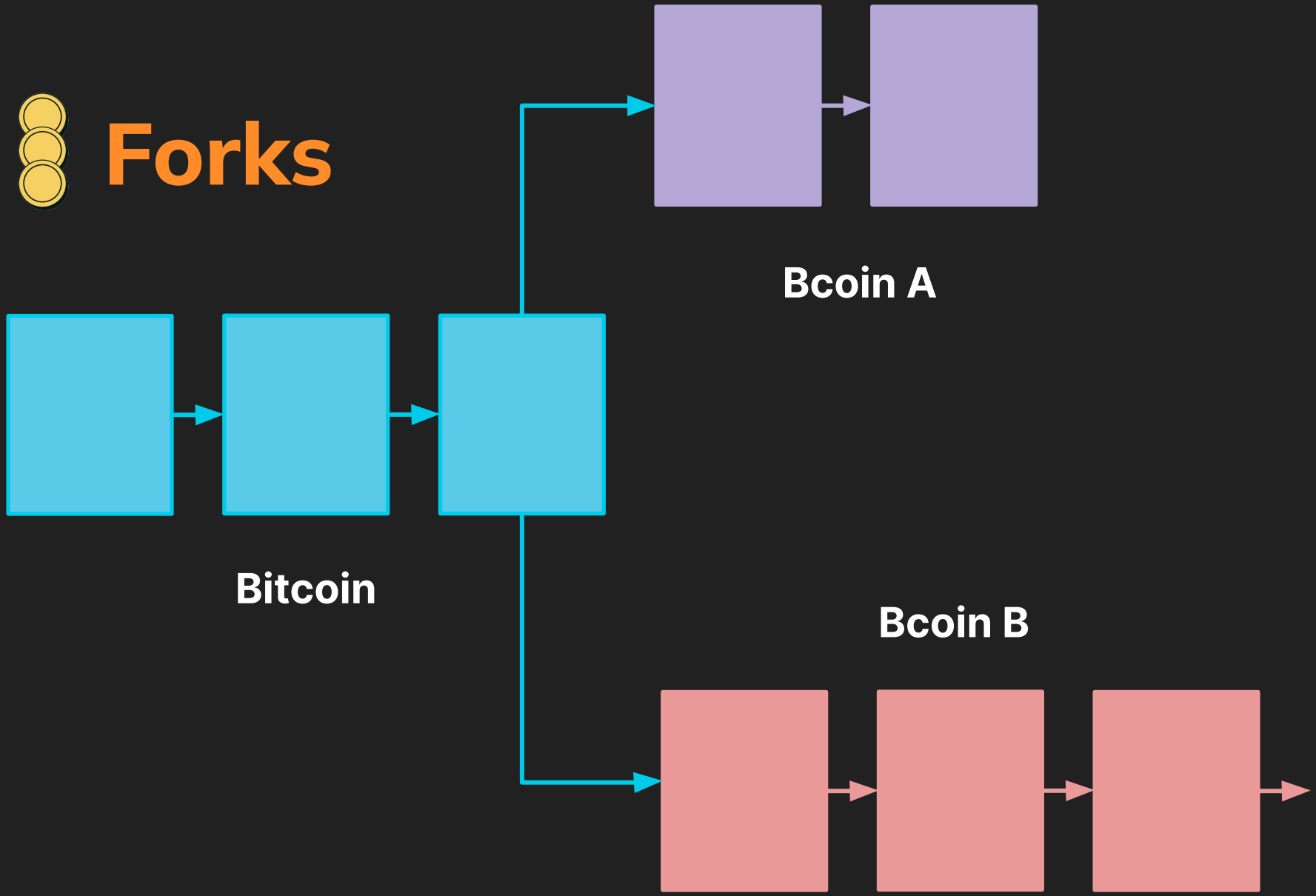
Contribuidores y Desarrolladores Core

- Wladimir J. van der Laan
- Marko Falke
- Michael Ford
- Jonas Schnelli
- John Newberry

 ¿ Centralización Core Devs ?

- Lideran el desfile.
- Cualquier persona puede usar el software y agregar nuevas propuestas.
- Si un cambio no es bien recibido por la comunidad:
  - Los usuarios pueden hacer un fork sobre las reglas.
  - Empoderamiento.

Forks



Tomado de: XKCD: Workflow

Usos Bitcoin Core

- Nodo validador en la red P2P.
- Implementación de referencia de una billetera.
- Interfaz Programática (RCP via HTTP o Cli).

Módulo 1.1:
Arquitectura

P2P

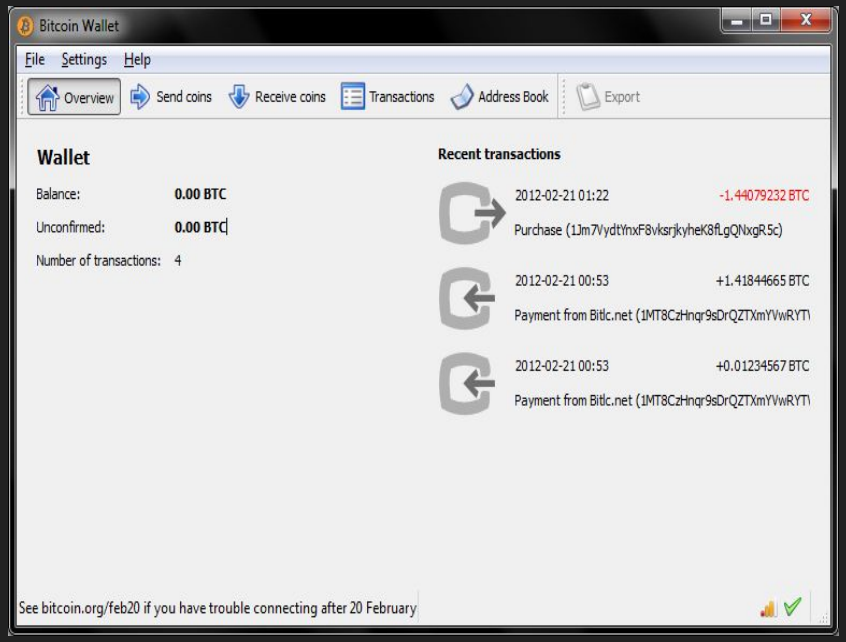
- Bitcoin forma una red TCP de mensajería.
  - src/protocol.h
- Cada nodo tiene pares/vecinos con los cuales puede intercambiar información.
- Protección DoS.
- SPV.

RPC/HTTP

- Interacción programática con Bitcoind
  - Consultas.
  - Uso de billeteras externas.
  - Mineros construyen bloques.
  - *bitcoin-cli* permite acceder a estar interfaz a través de la línea de comandos.

Qt

- Interfaz gráfica que expone:
  - Estadísticas básicas.
  - Consola RPC.
  - Funcionalidades de billetera.



ZMQ

- Publica notificaciones en un socket cuando recibe:
  - Un nuevo bloque.
  - Una nueva transacción.
- Útil para otros aplicativos para realizar acciones sobre esos eventos (Lightning Network).

Concurrencia

- Bitcoin Core realiza tareas simultáneas.
- Hilos.
- Interfaz de Validación.

Concurrencia

Objetivo	# de hilos	Tarea
Verificación de scripts	nproc o 16	ThreadScriptCheck()
Carga de bloques	1	ThreadImport()
Responder a llamados RPC	4	ThreadHTTP()
Inicializar conexiones de red	1	ThreadOpenConnections()

Secciones

- Subsistemas existentes en Bitcoin Core.
- Procedimientos necesarios para la operación de Bitcoin.

Nombre	Descripción
net	Administra la red a través de sockets y el monitoreo del nodo.
net_processing	Enruta mensajes P2P.
validation	Define cómo validar nuestro estado (cadena, mempool).
txmempool	Estructura de datos para mempool.
coins & txdb	Interfaz para el conjunto de UTXO.
script/	Ejecución de Script.
consensus/	Parámetros de consenso, raíces de Merkle.
policy/	Estimación de tarifa, reemplazo por tarifa.
indexes/	Creación de índices (txindex).
wallet/	Base de datos de billetera.

Almacenamiento

- Bitcoin almacena alguna información en archivos .dat
  - .dat: Contiene los bytes de una estructura de datos serializada.
    - serialize.h
  - tree ~/bitcoin/testnet3

Archivos .dat

- **blocks/blk.dat:** Información serializada de bloques.
- **blocks/rev.dat**: información “desechada”, UTXOs añadidas y removidas por un bloque.
- **mempool.dat:** lista serializada de componentes del mempool.
- **peers.dat:** pares serializados.
- **banlist.dat:** Ips baneadas por el nodo.

leveldb

- Almacenamiento ordenado de tipo llave-valor.
- Permite escrituras en volumen y snapshots.
- Disminuye posibles bugs.

Contenido leveldb

- **blocks/index**: Contiene todos los bloques que el nodo ha visto.
- **chainstate**: Conjunto de UTXO.

berkeleydb

- BerkeleyDB es similar a leveldb.
- Usada principalmente para la implementación de billetera.
- En proceso de ser reemplazada por SQLite.
- Línea de tiempo propuesta.

Contenido berkeleydb

- **Wallet:**
  - Wallets/wallet.dat: archivo que contiene la billetera.

Estructuras de datos

- CBlockheader
- CBlock
- CBlockIndex
- CChainstate

Módulo 1.2:
Contribución

Contribuir a Bitcoin Core

- Revisa y realiza pruebas sobre el código.
- Lee la documentación.
- Good First Issue.

Bitcoin Core PR Review Club

bitcoin / doc /

Qt 5? Qt2 is correct good first issue

#19076 opened on May 26, 2020 by slowdown2016

Canales

- Libera.chat
  - web.libera.chat #bitcoin-core-dev
- Discusiones en Github.
- Lista de correos.
- PR Review Club.
- Bitcoin Stack Exchange.

Contribuir a Bitcoin Core

- Embajador.
- Traductor.
- Mejoras en la documentación.
- Pruebas.
- Reporte de *issues*.
- Participar en los foros, discusiones, etc.

Cuellos de Botella

- No es la discusión sobre el tamaño del bloque.
- No es el consumo energético.
- No es su algoritmo de consenso.

- **Falta de desarrolladores calificados.**

Habilidades Core Dev

- Conocimiento de C++ y Python.
- Git.
- Instalar y remover paquetes de tu sistema.
- Motivación real, no es algo imposible.

Habilidades Apps Dev

- Conocimiento de algún lenguaje de programación.
- Git.
- Entender cómo funciona Bitcoin.
- Capacidad de crear aplicaciones enfocadas en el usuario que permitan mayor adopción.

Módulo 1.3: Corre tu propio nodo

¿ Por qué correr tu propio nodo ?

- No depender de terceros.
- No confíes, verifica.
- Validas transacciones, contribuyes al sistema y disminuye la necesidad de confianza.
- Disminuyes el riesgo de centralización en unos cuantos participantes de la red.
- Disminuyes el riesgo de censura.

Ventajas Full Node

- Nadie pudo crear dinero de la nada.
- Nadie puede gastar salidas sin la llave privada asociada.
- Nadie puede gastar una salida 2 veces.
- Nadie puede violar las reglas del sistema.

Confianza en otros

- SPV Node:
  - Fe ciega en los mineros.
  - No puede validar transacciones, bloques y la cadena generada.
- Servicios Centralizados:
  - Basados en la confianza que defina el portal, billetera web, etc.
  - No tienes las llaves, no son tus monedas.

Módulo 1.4: Corre tu propio nodo en ubuntu

Módulo 2:
Script

Contratos

- Encuentro donde 2 o más se unen con intenciones compartidas.
- Vínculo-Obligación-Penalidad.
- Libertad.

Contratos inteligentes

- Conjunto de promesas en formato digital, incluyendo protocolos para cumplir con estas.
- Observabilidad.
- Verificabilidad.
- Privacidad.
- Exigibilidad.

Teorema de Post

- $\mathbf{F}$ es el conjunto de funciones booleanas, el sistema de esas funciones $\{f^1, f^2, f^3 \ldots\}$ es completo si cualquier función booleana puede desplegarse del mismo.
- Turing completo vs. Turing incompleto.
- Testigo.
- Verificación vs. Computación.

Definimos una moneda electrónica como una cadena de firmas digitales. Cada dueño transfiere la moneda al próximo al firmar digitalmente un hash de la transacción previa y la clave pública del próximo.

Script

- No es nombrado en el whitepaper.
- EvalScript.
- Máquina de pila que evalúa y retorna validez.

Script

- Soporte para todo tipo de transacción.
- Cada nodo solo necesita entender la transacción para evaluarla.
- Predicado.

Script: historia

- Scriptsig
- EvalScript
- P2SH
- Segregated Witness
- Taproot

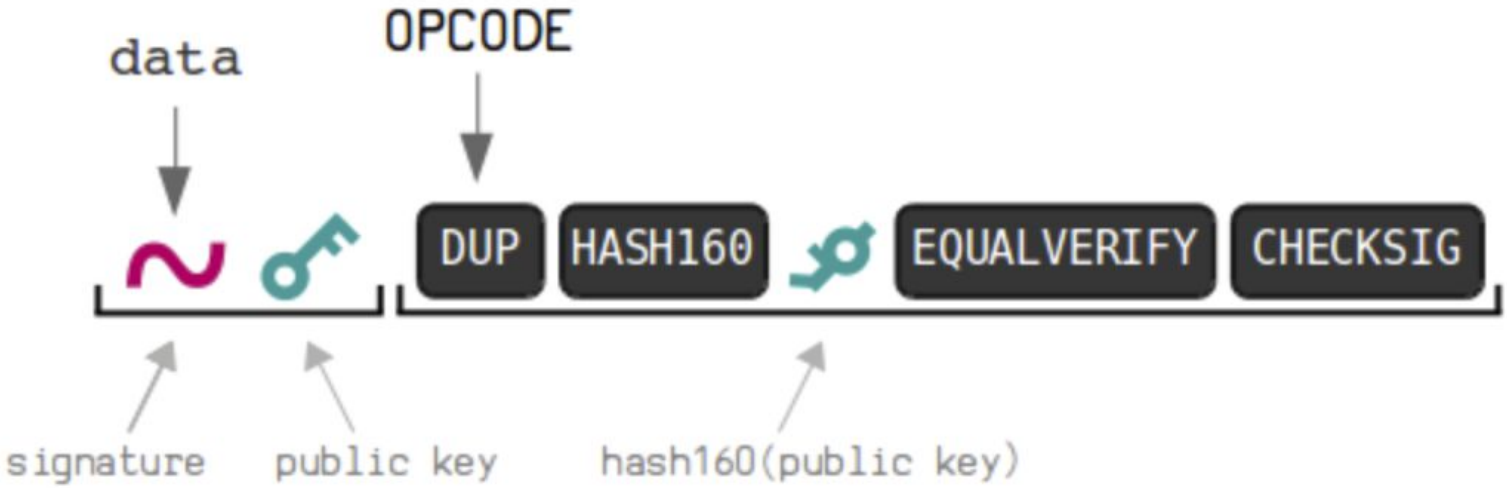
¿ Qué es Script ?

- Lenguaje de programación usado como mecanismo que bloquea salidas.
- Un script de bloqueo se agrega en cada salida.
- Un script de desbloqueo debe proveerse para desbloquear una salida.

Componentes

- Datos (LIaves públicas y firmas).
- OPCODES: funciones que permiten operar los datos.

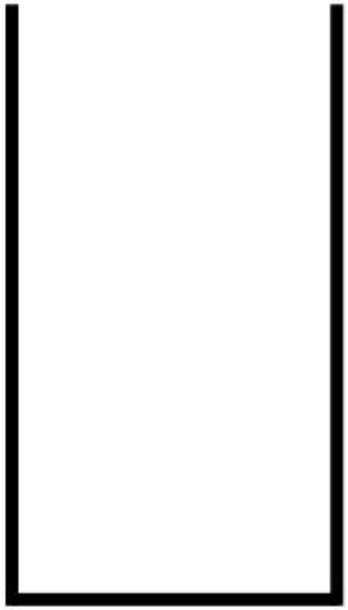
Componentes



Tomado de: https://learnmeabitcoin.com/technical/script

Cómo es ejecutado

- Siempre de izquierda a derecha.
- Los datos siempre se empujan a la pila
- OPCODES pueden sacar elementos, y opcional, empujar.
- Válido si la cima y único elemento en la pila es 1 o más.



Stack

Tomado de:
https://learnmeabitcoin.com/technical/script

¿ Dónde lo encuentras ?

- Script que bloquea en cada salida que se crea en una transacción.
- Script que desbloquea cada vez que quieres usar una salida como entrada en una nueva transacción.
- Siempre se ejecuta el script de desbloqueo primero que el script de bloqueo.

¿ Por qué lo usamos ?

- ¿ Por qué no usar una llave pública y la firma solamente y no usar OPCODES ?
- Podemos crear distintos tipos de bloqueos con distintas combinaciones, flexibilidad.

Ejemplos

- Matemático

  Suma: ‘[1 2 OPP_ADD]’

  Resta: ‘[3 2 OPP_ADD 4 OPP_SUB]’

- P2PKH

  <signature> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

Scripts Estándar

- P2PK
- P2PKH
- P2MS
- P2SH
- NULL

Módulo 2.1: Ejecuta Scripts

Conclusión