ln-paywall

Go middleware
for monetizing your API
on a per-request basis
with the Bitcoin Lightning Network

Contents

The Pain

The Pain

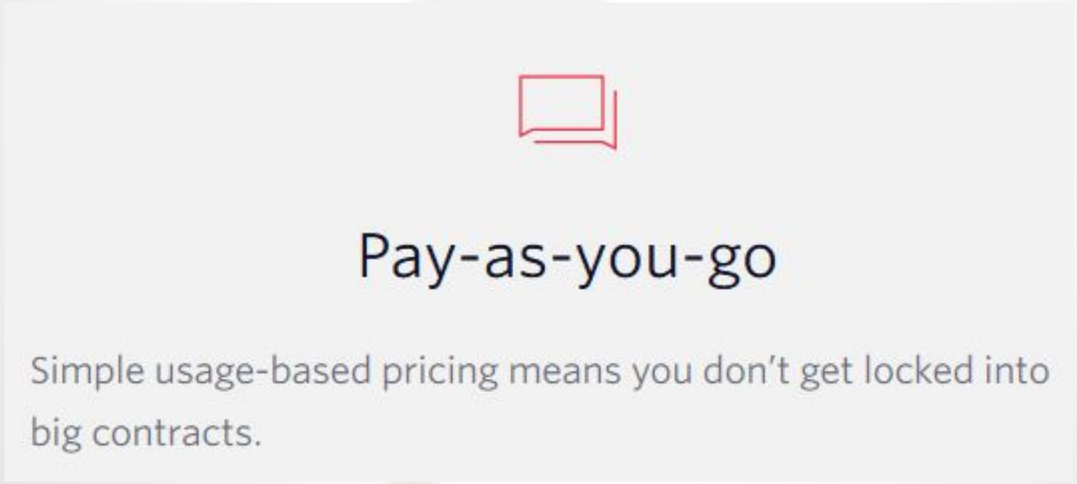
“Current API paywalls are a pain in the ass”

- Philipp

- Centralized payment method (PayPal, Bank)
  - Can shut down or deny service
- High fees for payments (~ $0.30)
- Have to keep track of API users
  - => Privacy concerns, data breaches / leaks
- No real per-request billing

The Pain

Example: Twilio

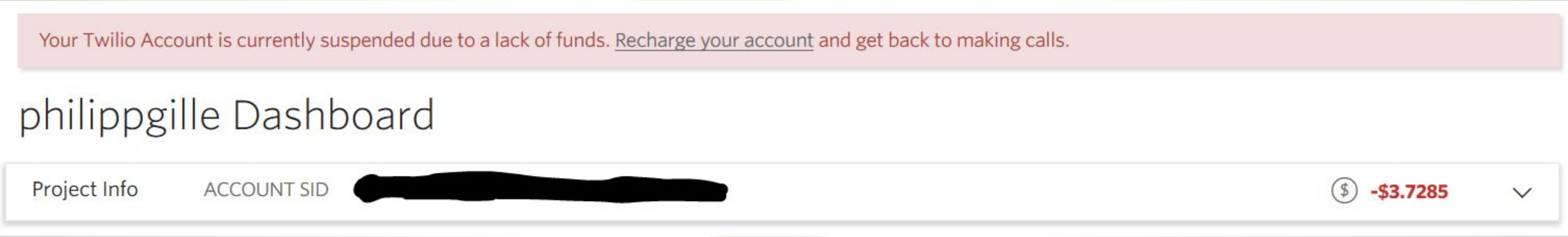


Pay-as-you-go

Simple usage-based pricing means you don't get locked into big contracts.

The Pain

Example: Twilio



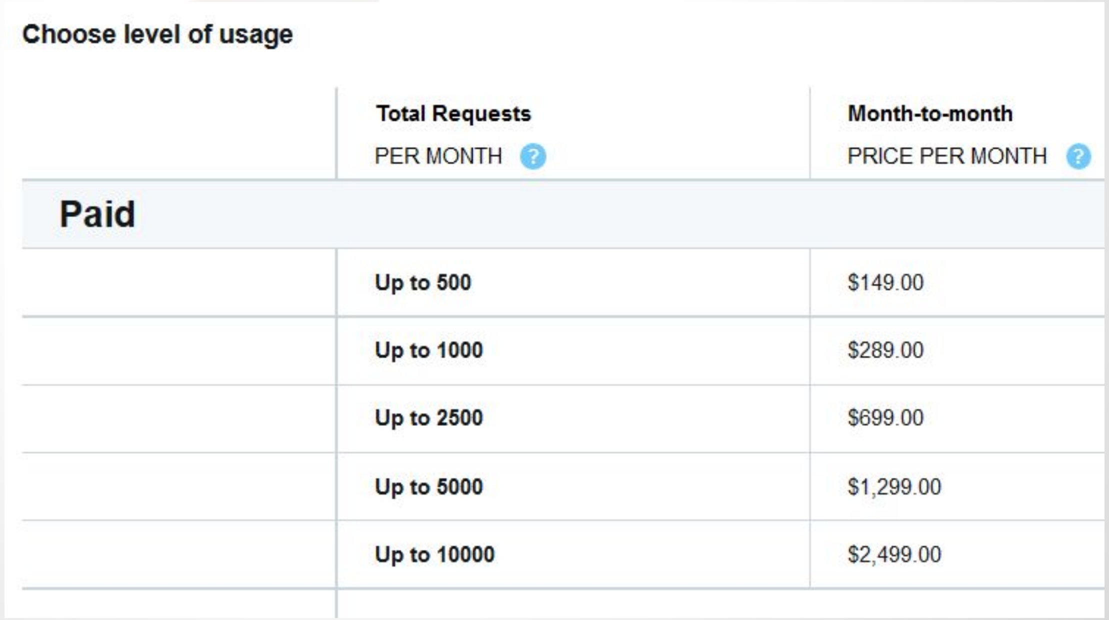
Your Twilio Account is currently suspended due to a lack of funds. Recharge your account and get back to making calls.

philippgille Dashboard

Project Info ACCOUNT SID [redacted] -$3.7285

The Pain

Example: Twitter



Choose level of usage

	Total Requests PER MONTH	Month-to-month PRICE PER MONTH
Paid		
	Up to 500	$149.00
	Up to 1000	$289.00
	Up to 2500	$699.00
	Up to 5000	$1,299.00
	Up to 10000	$2,499.00

The Solution

The Solution

Cryptocurrencies in general?

- p2p
- No expensive middlemen
- No for-profit company
- No legacy banking systems
- ...

The Solution

Cryptocurrencies in general?

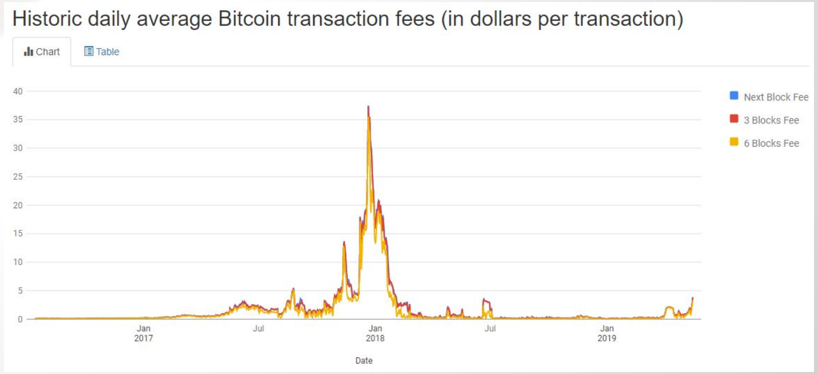
Nope.

The Solution

Cryptocurrencies in general?

Nope:

- Long confirmation times
  - Bitcoin: 10m/conf; 6 conf = safe
- High transaction fees
  - Bitcoin: Depends. Currently ~$3
- Doesn’t scale
  - Bitcoin: 7 tx/s



The Solution

Lightning Network

The Solution

Lightning Network:

- Second layer on top of the Blockchain
- Routed “payment channels”
  - Enabled via the underlying Blockchain’s *smart contracts*
- Near-instant microtransactions (no *mining*)
- Extremely low fees
  - E.g. ACINQ node: $0.0008 + 0.0001%
- Higher privacy (no on-chain traces, *onion routing*)
- No compromise on safety

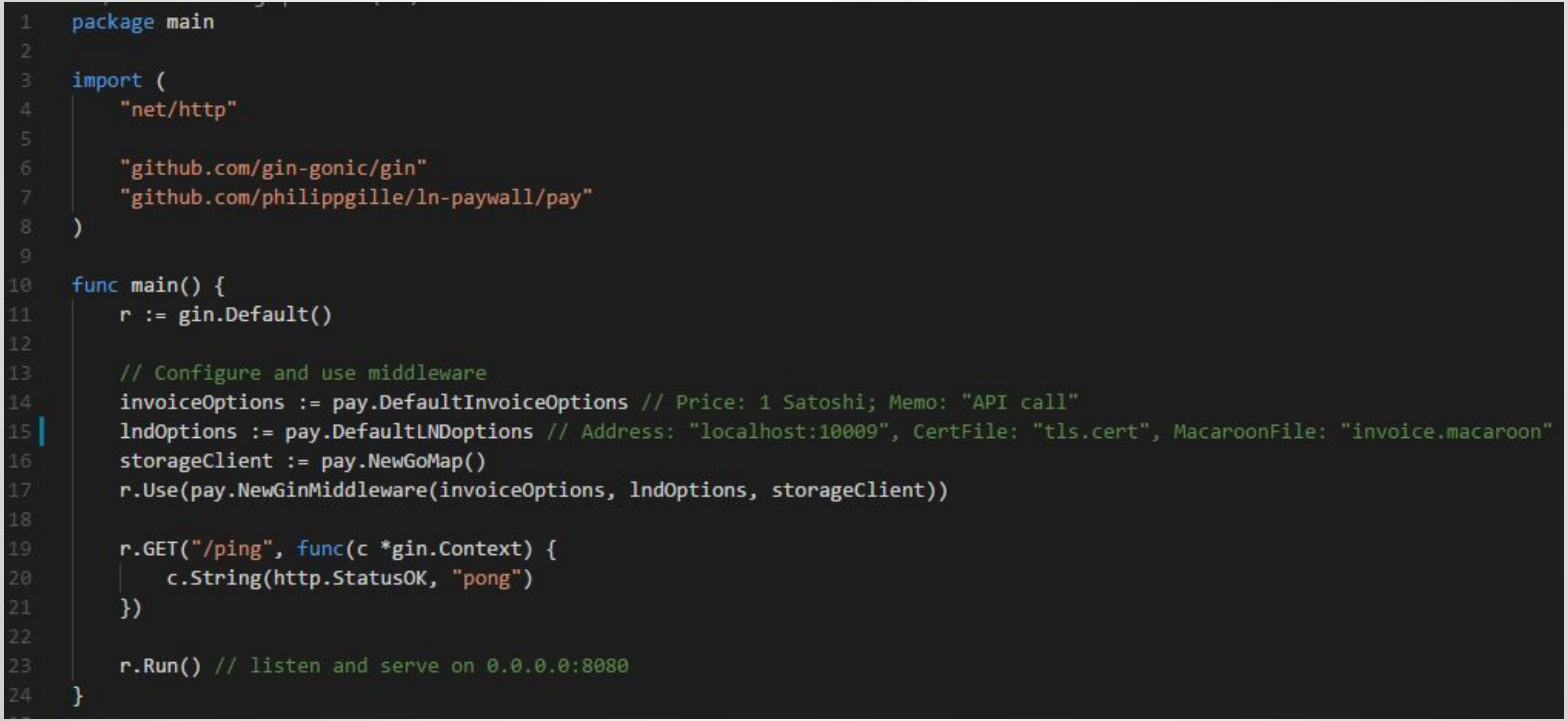
DEMO

DEMO

- lightning.ws:
  - curl -v https://api.lightning.ws/translate?text=Hallo%20Welt&to=en
    - curl -H “x-preimage: …” ...
  - https://staging.lightning.ws

- Others:
  - https://testnet.yalls.org
  - https://starblocks.acinq.co/
  - …

DEMO



```go
package main

import (
	"net/http"

	"github.com/gin-gonic/gin"
	"github.com/philippgille/ln-paywall/pay"
)

func main() {
	r := gin.Default()

	// Configure and use middleware
	invoiceOptions := pay.DefaultInvoiceOptions // Price: 1 Satoshi; Memo: "API call"
	lndOptions := pay.DefaultLNDoptions // Address: "localhost:10009", CertFile: "tls.cert", MacaroonFile: "invoice.macaroon"
	storageClient := pay.NewGoMap()
	r.Use(pay.NewGinMiddleware(invoiceOptions, lndOptions, storageClient))

	r.GET("/ping", func(c *gin.Context) {
		c.String(http.StatusOK, "pong")
	})

	r.Run() // listen and serve on 0.0.0.0:8080
}
```

Behind the Scenes

Bitcoin Basics

Behind the Scenes - Bitcoin Basics

- Creator:
  - “Satoshi Nakamoto” - Unknown identity
  - Vanished when a contributor wanted to show Bitcoin to the NSA
  - Emails, forum posts etc.: https://satoshi.nakamotoinstitute.org/
- Whitepaper:
  - **2008**-10-31
  - "Bitcoin: A Peer-to-Peer Electronic Cash System"
  - http://bitcoin.org/bitcoin.pdf
- First block in the Blockchain:
  - **2009**-01-03
  - Includes message:
    - “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.”

Behind the Scenes - Bitcoin Basics

- “Bitcoin” is …
  - A cryptocurrency
    - “*Alice gives Bob one Bitcoin*”
  - A blockchain
    - “*The Bitcoin blockchain currently consists of 537,000 blocks*”
  - A p2p protocol
    - Like HTT**P** is used between web browsers and servers
  - A software (for running a node)
    - “Official” implementation: Bitcoin Core / bitcoind
    - Others: btcd, libbitcoin

Behind the Scenes - Bitcoin Basics

- Bitcoin can be viewed from different points of view:
  - Ideological / political
    - Cypherpunk: Decentralized, anonymous, electronic payments
      - The roots of Bitcoin!
    - Crypto-anarchist: Against banks, the state, taxes
  - Financial
    - Trader: “Sick gains”
    - Remittance: Cheap, fast international transfers (no middlemen)
  - Criminal
    - Drug dealer: Money laundering
  - Practical
    - Unbanked (2 billion): Bank account in your pocket
    - **Developer**: Revolutionary **technology**; **“programmable money”**

Behind the Scenes - Bitcoin Basics

Tech:

- Transaction
  - Smart Contract
- Block
  - Mining / Proof of work

Behind the Scenes - Bitcoin Basics

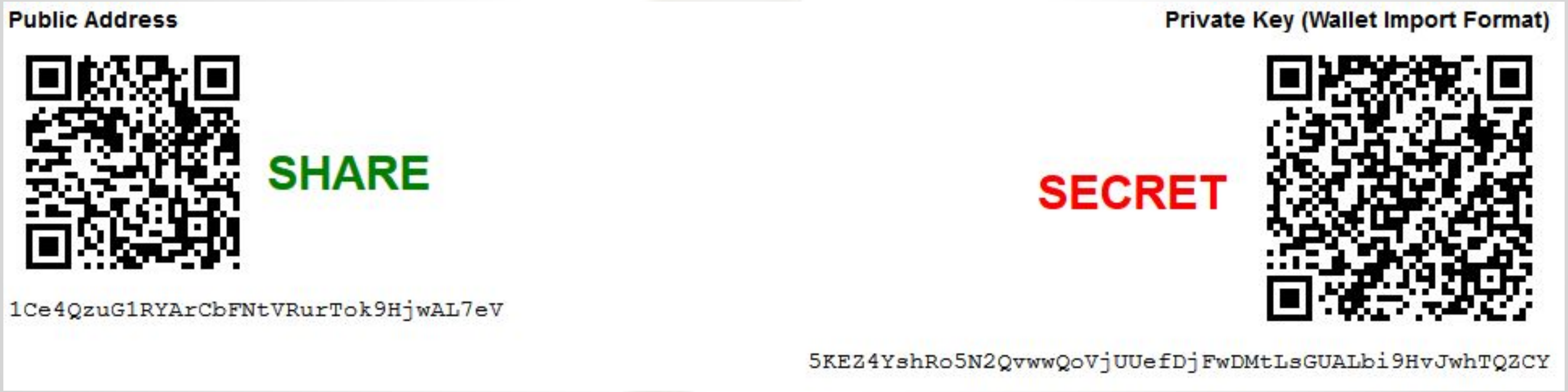
Transaction: DEMO

Behind the Scenes - Bitcoin Basics

Transaction:

View in Blockchain explorer: https://blockstream.info/

Behind the Scenes - Bitcoin Basics



Public Address

SHARE

1Ce4QzuG1RYArCbFNtVRurTok9HjwAL7eV

Private Key (Wallet Import Format)

SECRET

5KEZ4YshRo5N2QvwwQoVjUUefDjFwDMtLsGUALbi9HvJwhTQZCY

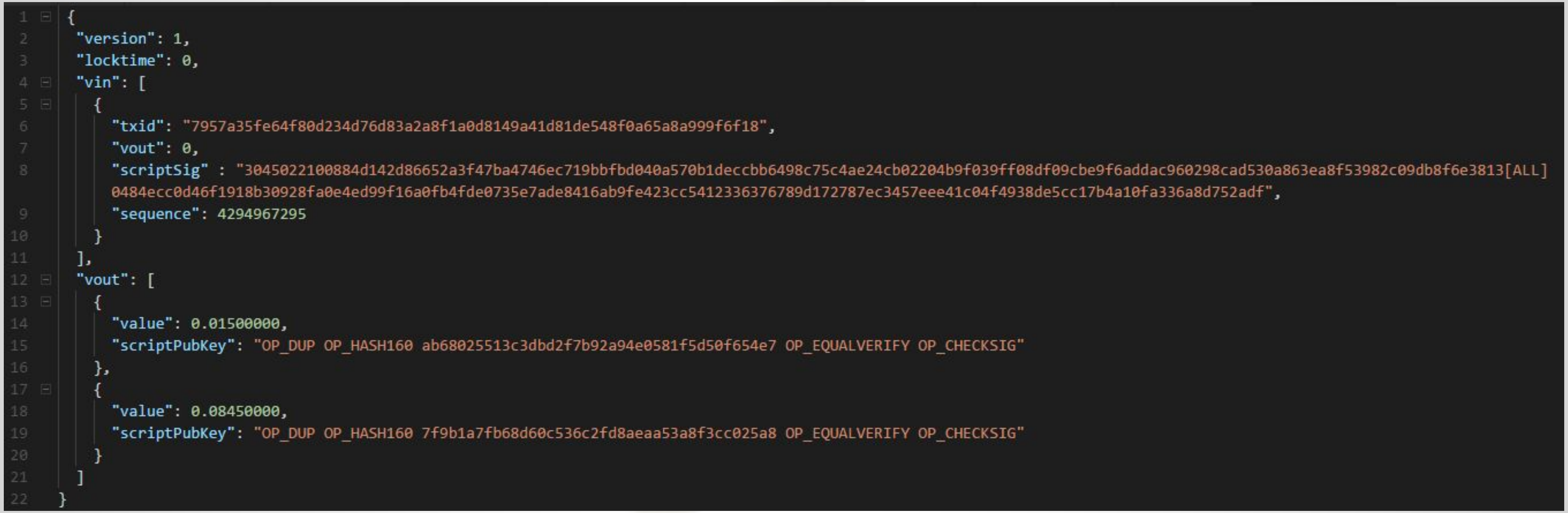
Behind the Scenes - Bitcoin Basics

1. Random 256 bit number
   - E.g. SHA256(x)
2. Calculate public key
   - Elliptic curve: secp256k1; algorithm: ECDSA
3. Calculate Bitcoin address
   - RIPEMD160(SHA256(Public key))
     - => **Public key hash**
   - Encode with Base58Check
     - => Bitcoin address

Behind the Scenes - Bitcoin Basics

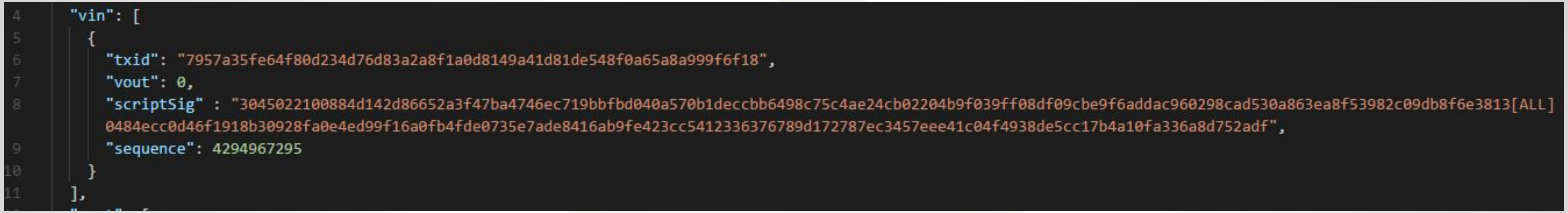
Transaction?

Behind the Scenes - Bitcoin Basics



```
{
 "version": 1,
 "locktime": 0,
 "vin": [
   {
     "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
     "vout": 0,
     "scriptSig" : "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
     0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
     "sequence": 4294967295
   }
 ],
 "vout": [
   {
     "value": 0.01500000,
     "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
   },
   {
     "value": 0.08450000,
     "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG"
   }
 ]
}
```

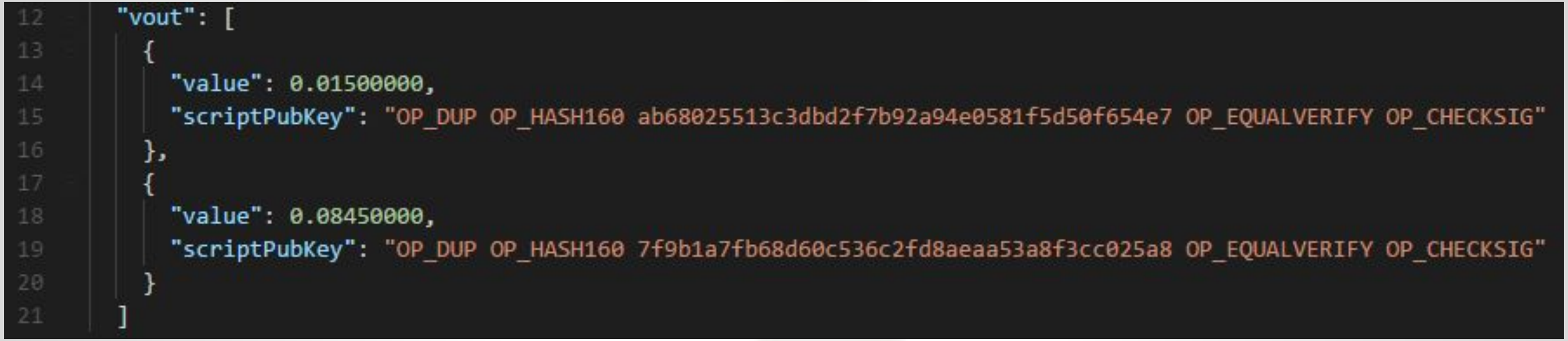
Behind the Scenes - Bitcoin Basics



```
  "vin": [
    {
      "txid": "7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" : "3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
      0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
```

- txid: References the tx that contains the UTXO being spent
- vout: Index of the UTXO
- scriptSig: Signature + public key
  - Satisfies the conditions placed on the UTXO
    - Unlocks the UTXO for spending
    - Proof of ownership

Behind the Scenes - Bitcoin Basics

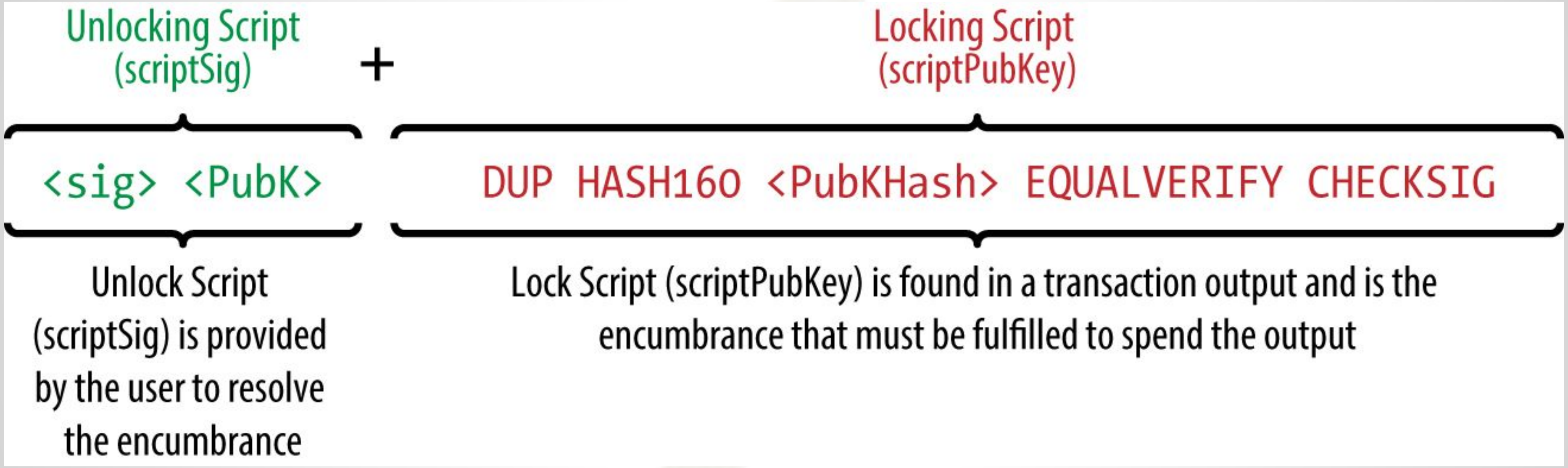


```
"vout": [
  {
    "value": 0.01500000,
    "scriptPubKey": "OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
  },
  {
    "value": 0.08450000,
    "scriptPubKey": "OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG"
  }
]
```

- UTXO = “Unspent transaction output”
  - “Alice owns 1 Bitcoin” =

    Alice’s wallet has detected 123 UTXOs that can be spent with the keys in that wallet.
  - Here: One “change” UTXO, one “normal” UTXO (= actual payment)
- scriptPubKey: “locking script” / “puzzle”
  - Determines the conditions required to spend the output

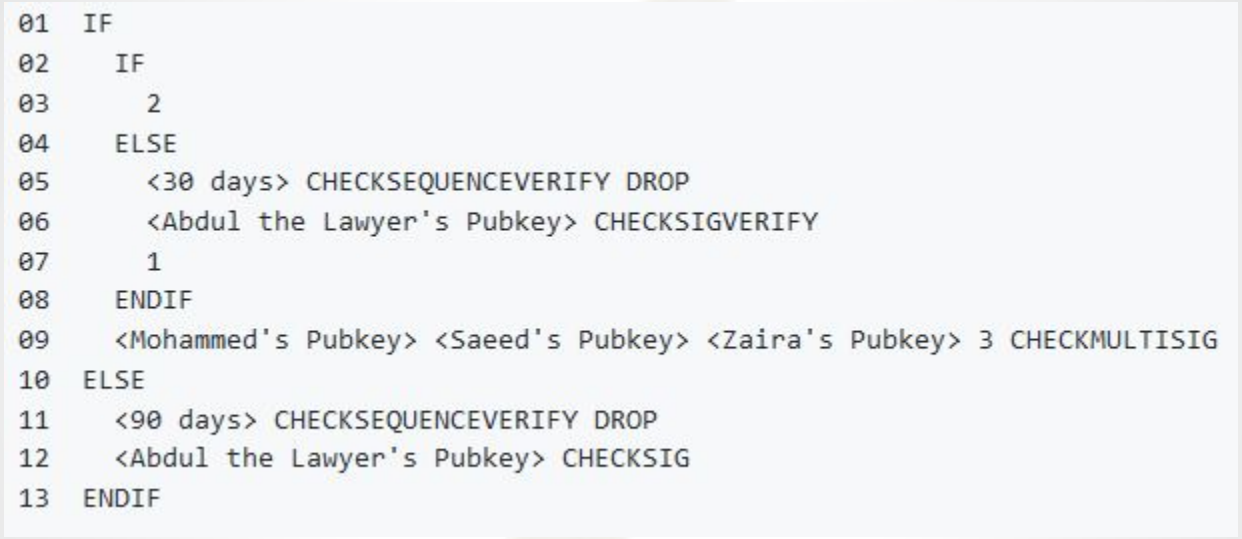
Behind the Scenes - Bitcoin Basics


Unlocking Script (scriptSig)
+
Locking Script (scriptPubKey)
<sig> <PubK>
DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG

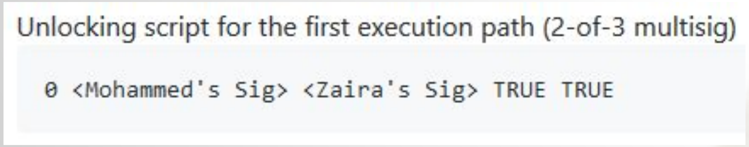
Unlock Script (scriptSig) is provided by the user to resolve the encumbrance

Lock Script (scriptPubKey) is found in a transaction output and is the encumbrance that must be fulfilled to spend the output

Behind the Scenes - Bitcoin Basics

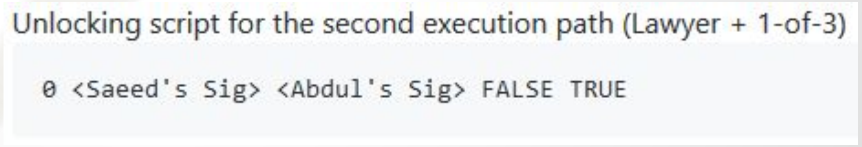


```
IF
  IF
    2
  ELSE
    <30 days> CHECKSEQUENCEVERIFY DROP
    <Abdul the Lawyer's Pubkey> CHECKSIGVERIFY
    1
  ENDIF
  <Mohammed's Pubkey> <Saeed's Pubkey> <Zaira's Pubkey> 3 CHECKMULTISIG
ELSE
  <90 days> CHECKSEQUENCEVERIFY DROP
  <Abdul the Lawyer's Pubkey> CHECKSIG
ENDIF
```



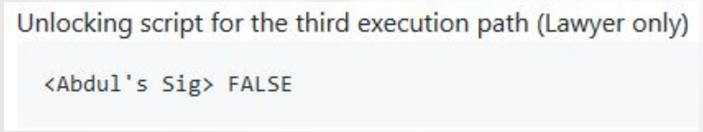
Unlocking script for the first execution path (2-of-3 multisig)

```
0 <Mohammed's Sig> <Zaira's Sig> TRUE TRUE
```



Unlocking script for the second execution path (Lawyer + 1-of-3)

```
0 <Saeed's Sig> <Abdul's Sig> FALSE TRUE
```



Unlocking script for the third execution path (Lawyer only)

```
<Abdul's Sig> FALSE
```

Behind the Scenes - Bitcoin Basics

Block, Mining?

Behind the Scenes - Bitcoin Basics

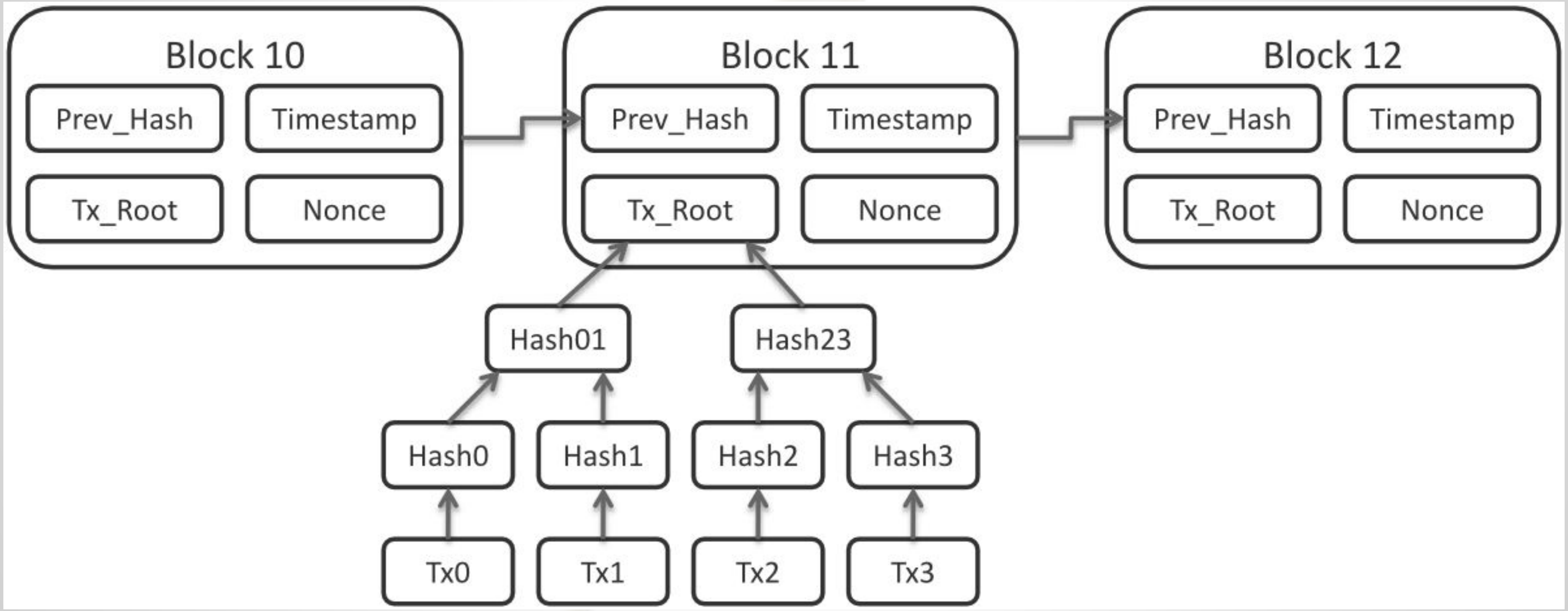
- Digital objects can always be copied
- Money must not be copyable
- Bank, PayPal?
  - *Centralized* ledger of payments
  - *Trusted* third party

=> How to achieve **scarcity**,

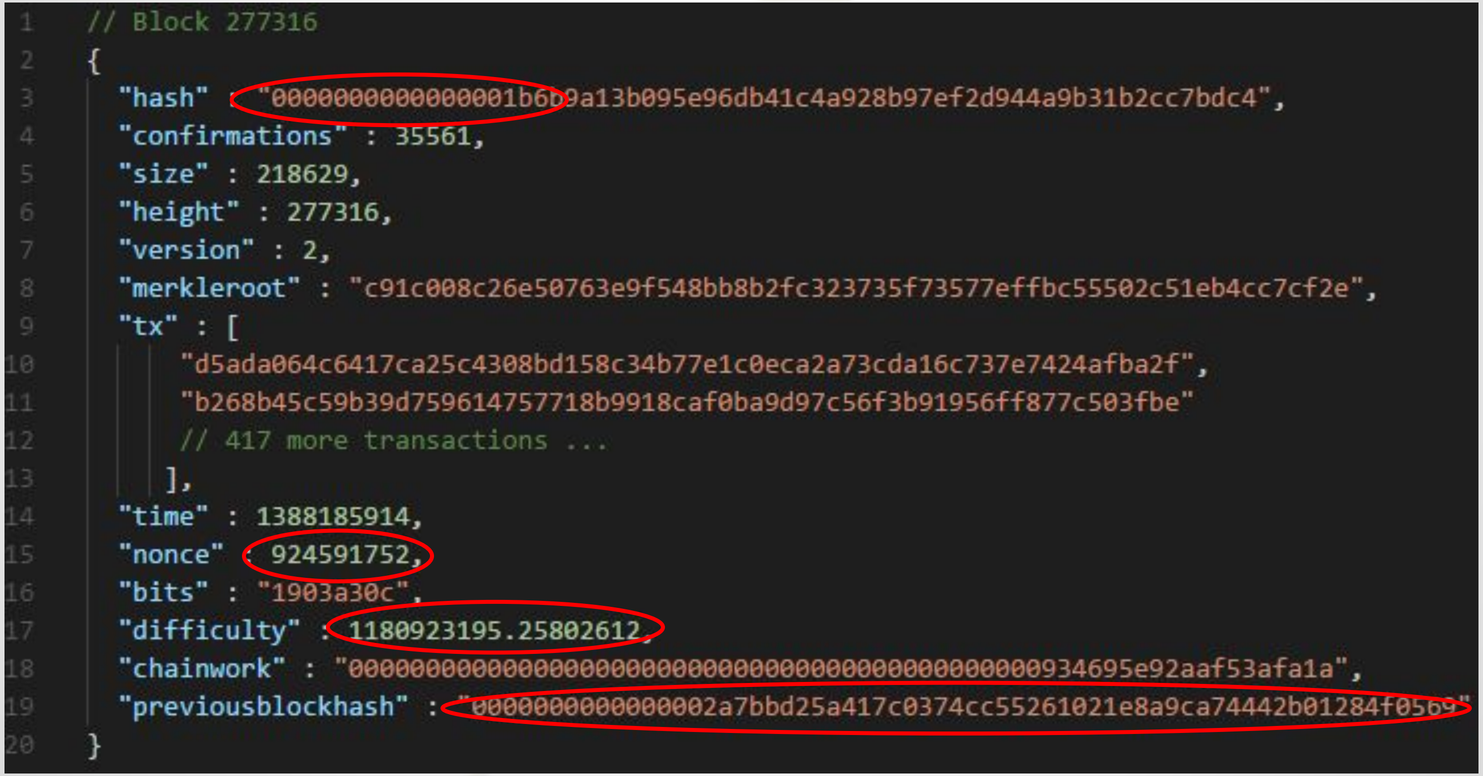
how to prevent a “**double spend**”

in a *decentralized, trustless* network?

Behind the Scenes - Bitcoin Basics



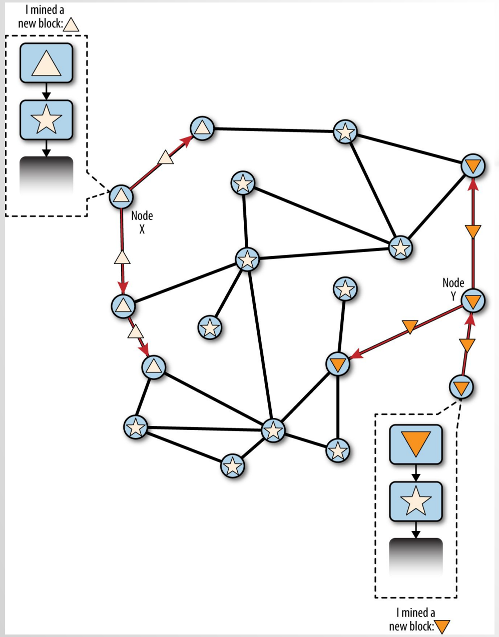
Behind the Scenes - Bitcoin Basics

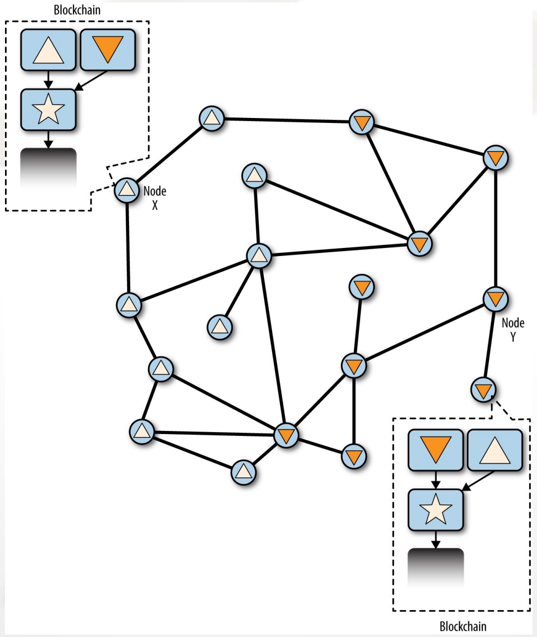


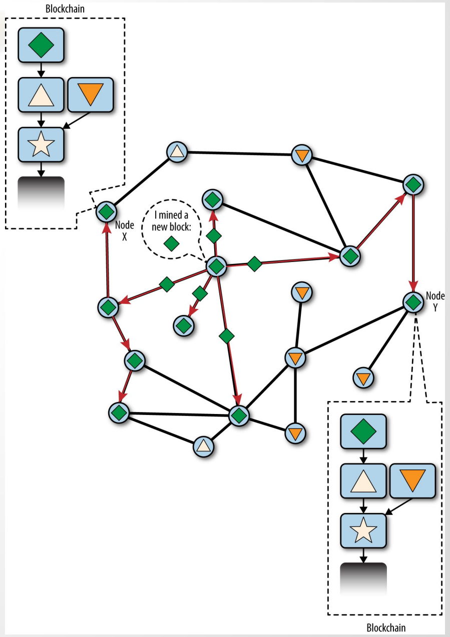
```
// Block 277316
{
  "hash" : "0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
  "confirmations" : 35561,
  "size" : 218629,
  "height" : 277316,
  "version" : 2,
  "merkleroot" : "c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e",
  "tx" : [
      "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
      "b268b45c59b39d759614757718b9918caf0ba9d97c56f3b91956ff877c503fbe"
      // 417 more transactions ...
    ],
  "time" : 1388185914,
  "nonce" : 924591752,
  "bits" : "1903a30c",
  "difficulty" : 1180923195.25802612,
  "chainwork" : "000000000000000000000000000000000000000000000934695e92aaf53afa1a",
  "previousblockhash" : "0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569"
}
```

Behind the Scenes - Bitcoin Basics

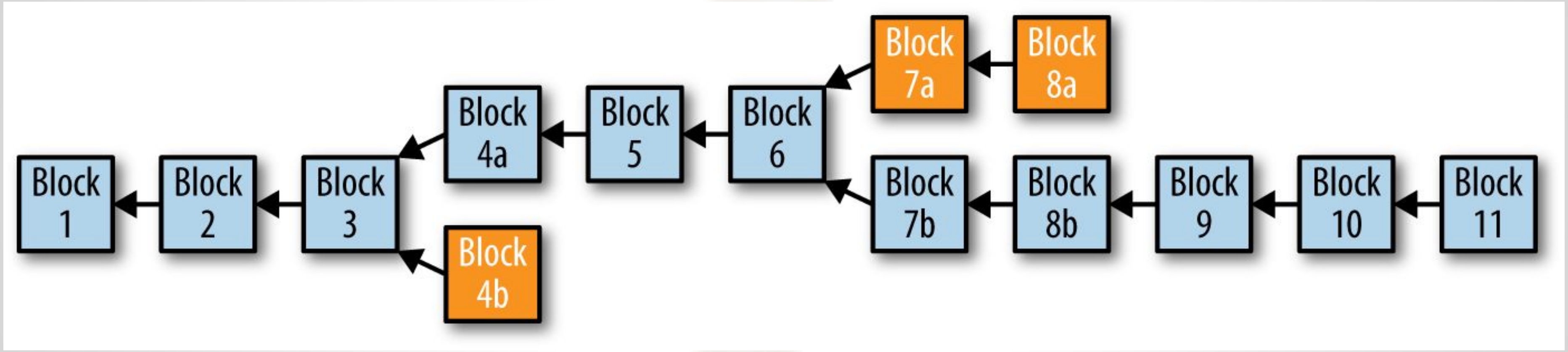
- What if a miner in China finds a valid block at the same time as a miner in the US?







Behind the Scenes - Bitcoin Basics



Payment Channels

Payment Channels

1. One on-chain tx, “funding tx”
   - 2-of-2 multisig from Alice and Bob
   - E.g. one input with 1 BTC each, 2 BTC output to multisig addr

2. Potentially thousands of *signed* **off-chain** tx, “commitment tx”
   - E.g. the 2 BTC UTXO as input, 0.9 BTC to Alice, 1.1 to Bob
   - Both parties could make the latest tx public at any time

3. Second / final on-chain tx, “**settlement** tx”
   - E.g. the 2 BTC UTXO as input, 0.5 BTC to Alice, 1.5 to Bob

Payment Channels

- Possible fraud:
  - Alice sends the first commitment tx on-chain
    - => She gets back 0.9 instead of 0.5 BTC
  - Alice doesn’t sign any commitment tx to Bob
    - => Bob’s funds are locked in the multisig forever

- Solution: Timelocks
  - E.g. OP_CHECKSEQUENCEVERIFY
    - => Tx can only be spent after some blocks
  - Each commitment tx has a shorter timelock
    - => Old tx can’t be broadcast before newer tx

Payment Channels

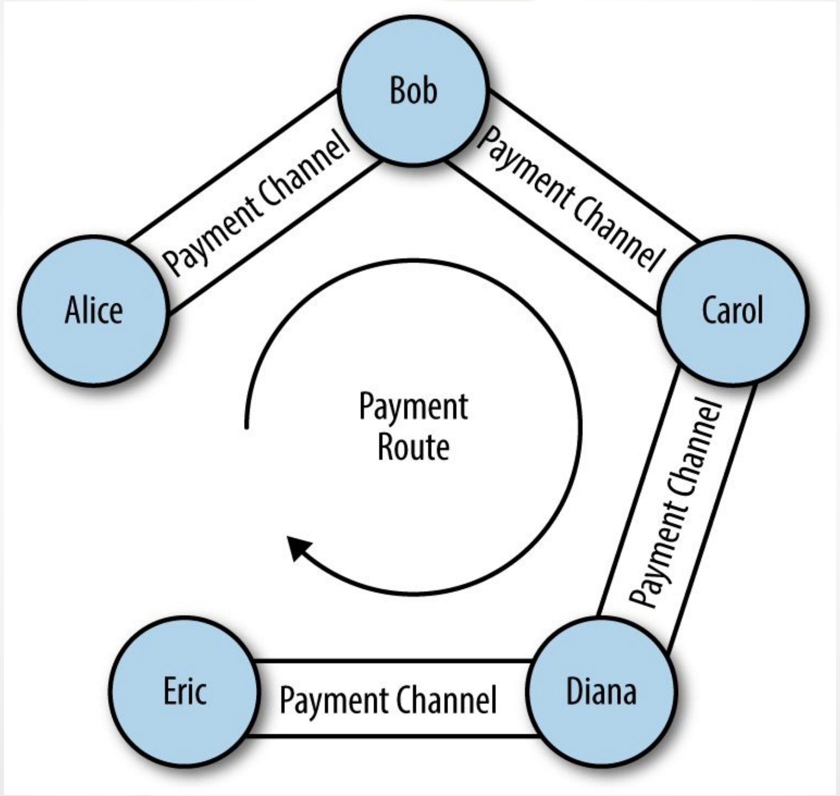
Disadvantages:

- Timelock
  - => Max channel age
- Lower timelock per tx
  - => Max number of tx per channel
- One channel (= one on-chain tx) to each party
  - Expensive
  - Not scalable
- Funding + settlement tx reveal payer and payee

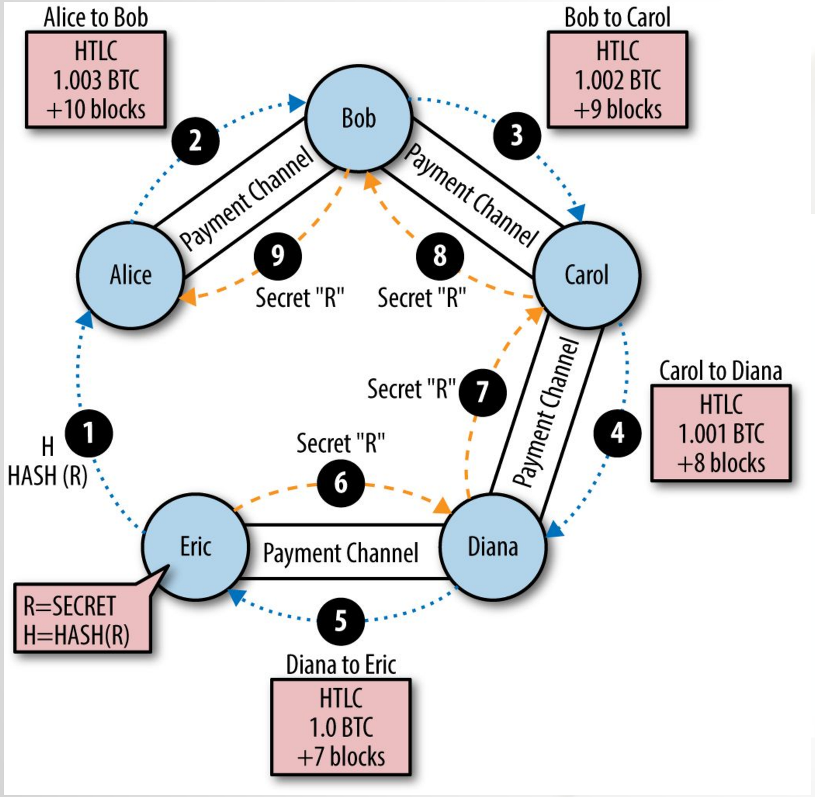
Lightning Network

Routed Payment Channels

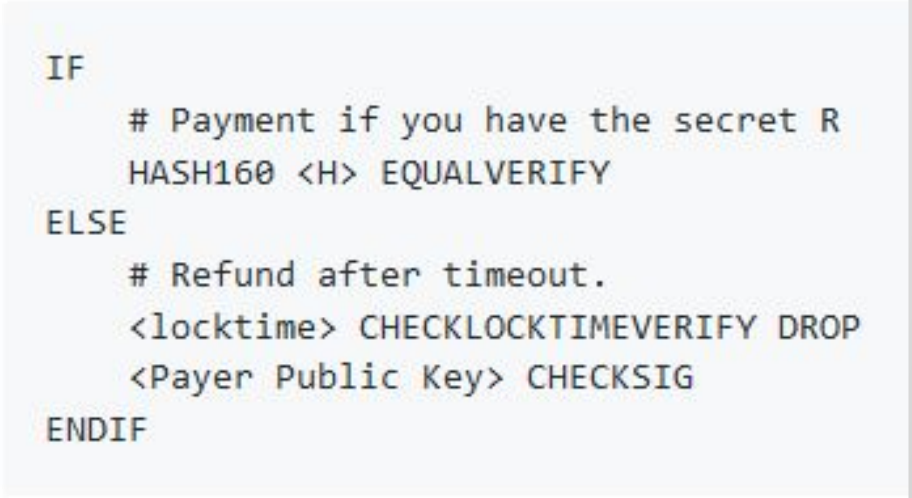
Lightning Network



Lightning Network



"HTLC": Hash Time Lock Contract


```
IF
    # Payment if you have the secret R
    HASH160 <H> EQUALVERIFY
ELSE
    # Refund after timeout.
    <locktime> CHECKLOCKTIMEVERIFY DROP
    <Payer Public Key> CHECKSIG
ENDIF
```

Lightning Network

- No max channel age
- Unlimited tx within a channel
- One channel can be enough to reach every other node
- Payments aren't revealed
  - (Channel from A to B, but A pays C)
- Onion routing
  - A routing node only sees the previous and next hop, not the payer or payee

Lightning Network

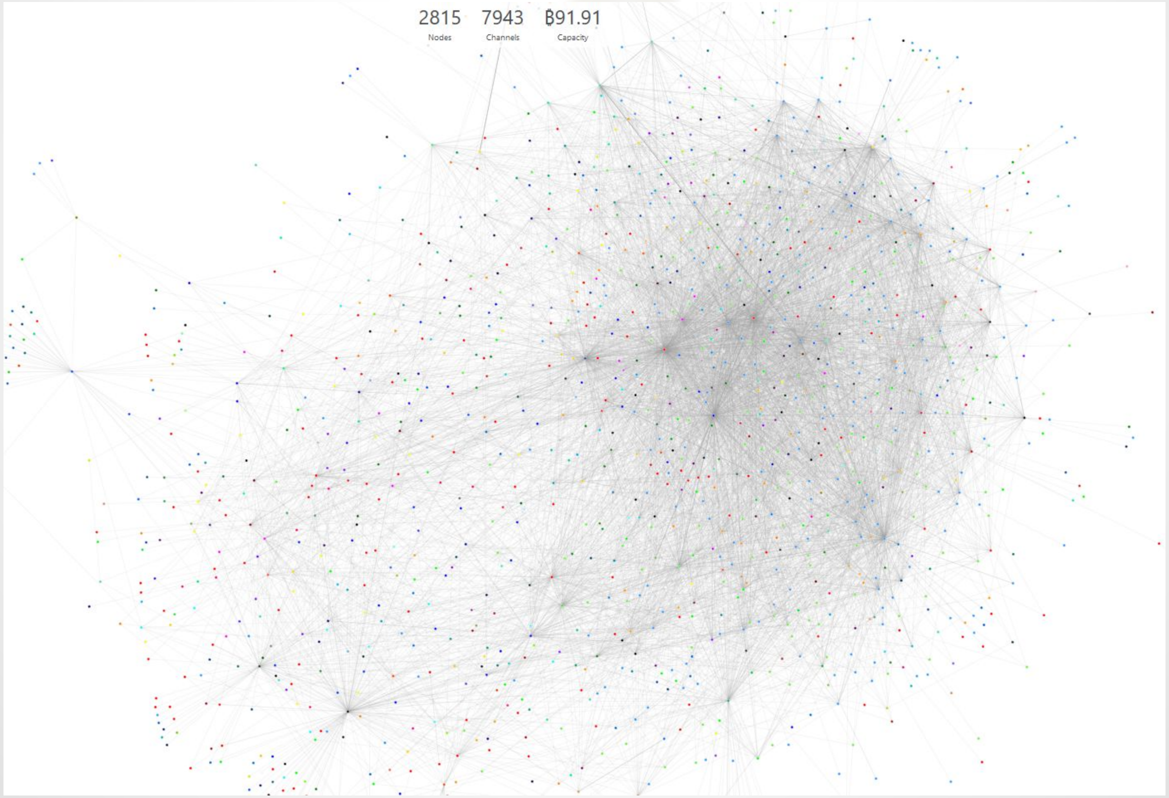
Current limitations being worked on:

- Max payment amount = max channel capacity
  - AMP (atomic multipath payments) will fix this
- Each channel requires a funding tx
  - “Channel factories” will fix this
- The amount of funds in a channel is fixed
  - “Splicing” allows changing the channel capacity in a single on-chain tx
- Two parties can only transact either on- or off-chain
  - “Submarine swaps” allow an off-chain payment to be received on-chain and vice-versa

Lightning Network

- Specification
  - https://github.com/lightningnetwork/lightning-rfc
- Multiple implementations
  - lnd (Go)
  - c-lightning (C)
  - Eclair (Scala)

Lightning Network



In-paywall

Code

ln-paywall

Deep dive:

https://github.com/philippgille/ln-paywall