

THE BITCOIN LAYER

ZINES!

<https://satsie.dev/zines>



@SATSIE

AUGUST 2024



booklets about bitcoin tech



open source

Always available to read online for free.

Thai translation made by the community!

Satsie's Pocket Guide to Taproot

Download the printable PDF here

Need help assembling? Check out these instructions.

What is the Taproot upgrade?

Taproot is a set of improvements that allow Bitcoin to be used in more scalable and private ways.

Activation date: November 2021
Block height: 709,632.

Taproot enables some cool features

Key and signature aggregation (MuSig)

If you have public keys A, B, and C, they can be combined into one. The same is true for the corresponding signatures.



This means complex multisignature spends can look like ones that only involve 1 key.

pg. 2

Batch signature validation

Validating digital signatures usually requires a lot of effort from a computer's CPU. Now transaction signatures can be grouped together and validated as one unit, instead of one by one.



Better privacy while spending

Bitcoin allows you to specify multiple ways to spend a coin. Prior to Taproot, all these ways had to be made public when the coin was spent. This is bad for privacy, especially for coins with unique spending rules, making them easy to identify.

pg. 3

Previous

Next

Satsie's Pocket Guide to BIPs

Download the printable PDF here

Need help assembling? Check out these instructions.

There is also a Thai version  made available by aekasitt.

But that's just the beginning...
How do BIPs get accepted and deployed into Bitcoin?

This is up to the community to decide. It's on the BIP author + its supporters to champion the idea. They need to spark interest and rally support.

"The BIP author is responsible for building consensus in the community and documenting dissenting opinions"
— from BIP-2

The beautiful thing about Bitcoin is it's decentralized. Anyone can make a BIP! But that also means no one has the authority to tell others how to allocate their time + resources. Nobody has to review your BIP, or even engage with it.

pg. 6

Sometimes you have to be patient. Some now popular BIPs sat around for months before anyone noticed them.

Why don't some BIPs move forward? Why don't they make it into Bitcoin?

Oftentimes, while the community may not necessarily oppose a BIP, it doesn't progress because not enough people are excited enough to put their own energy towards it.

Other factors include competing BIPs or general fatigue around certain BIPs.

Some BIPs end up getting withered or rejected.

Always remember, just because a BIP is proposed, it doesn't mean it's adopted :)

Previous



@SATSIE

SATSIE.DEV/ZINES

IMAGES: HERECOMESBITCOIN.ORG

short
and
sweet



SV2 can be used as is today, allowing miners and pools to enjoy better security and efficiency. Several pools support it or have announced plans to. You can even connect SV1 devices with SV2 pools! While SRI is still a work in progress, it means there are many opportunities to get involved. From documentation and testing, to code review and development, there is something for everyone.

IT'S A WIN FOR MINERS, IT'S A WIN FOR POOLS. IT REDUCES BARRIERS TO ENTERING THE MARKET, INCREASING PROFITABILITY, SECURITY AND EFFICIENCY. IT MAKES THE WHOLE BITCOIN NETWORK MORE ROBUST.

Want to read this? Visit satsie.dev/zines

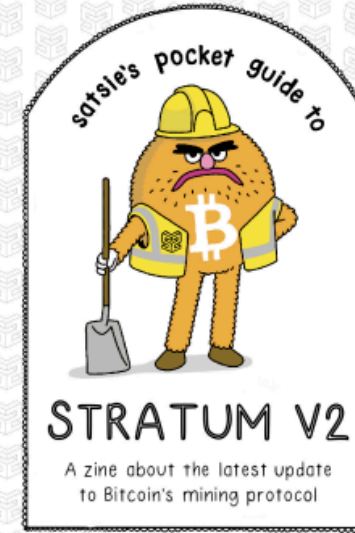


Miners can eliminate a source of revenue by declaring jobs themselves instead of pools. Future jobs are also more profitable, allowing devices to work on new jobs faster with resources that would otherwise be idle with SV1.

Performance improvements make mining more accessible to those with weaker internet connections, limited bandwidth, and less power. They lead to fewer empty blocks, higher efficiency, and higher profits!

Shortcomings

Miners pay their miners, and that the amount they pay is a significant amount are two issues that need to be addressed. Ideally, pools would be able to pay miners directly. Payouts would go straight to miners. There should also be much more transparency in how pools calculate the amount of work that each miner contributes.



A zine about the latest update to Bitcoin's mining protocol

APRIL 2024

By 2019 the mining landscape had drastically evolved. Late that year, the spec for Stratum V2 (SV2) was released.

Why is SV2 great?

Standardization

Despite widespread use, SV1 was never standardized. Parts of the spec are open to interpretation, resulting in inconsistent SV1 implementations that aren't guaranteed to be compatible. This defeats the purpose of a protocol!

In contrast, SV2 has a well defined spec maintained by an independent, open source working group. It's more developer friendly and has a large, growing community.

Security

In SV1, communications between miners and pools are unauthenticated and unencrypted. Messages are plain text and available for anyone to read! Miners are vulnerable to an attack called hashrate hijacking. When a miner sends the work they have done to a pool, an attacker can intercept the message and take all the credit.

In SV1, pools decide what transactions go into a block. If just a few of the largest pools collude, transaction censorship is possible. With the Job Declaration (JD) feature, SV2 flips this dynamic, letting miners pick the transactions. While pools can still reject blocks from miners, SV2 makes it easy for a miner to switch pools or solo mine. This incentivizes pools to avoid censorship, otherwise they risk losing miners.

JD also helps decentralize mining infrastructure. In addition to pools, miners and third parties can now construct block templates. This leads to more Bitcoin nodes, strengthening the network's peer-to-peer layer.

Stratum is a messaging protocol that miners and pools use to talk to each other

As a way to stabilize revenue, most Bitcoin miners join pools. Stratum is a protocol (set of communication rules) that allows individual mining devices ("devices") to talk to pools and the Bitcoin network. Stratum clients and servers form a communication layer above Bitcoin, facilitating the easy exchange of data between devices, Bitcoin nodes, and pools. It's used for things like connecting to, receiving jobs from, and submitting work to pools.

History

Slush (Marek Palatinus) proposed Stratum in late 2011. While initially intended for Electrum, a lightweight Bitcoin client, it was repurposed for mining. It quickly gained industry adoption and enjoyed years of success.

November 2019 The SV2 spec by Pavel Moravec, Jan Capek, Matt Corallo, + other industry experts is published.

January 2021 Work begins on a community based, open source implementation of SV2, The Stratum Reference Implementation (SRI).

Today

While SV2 has come a long way, development is far from over! There is still code in progress for things like monitoring, testing, and general usability. Even when SV2 is fully implemented, there are hurdles to overcome before complete adoption.

SV2 fixes this by adding authentication and encryption, a simple addition with a huge security impact.

Performance

Messaging: By making network messages smaller and less frequent, SV2 is more efficient and uses less bandwidth. Instead of JSON RPC, a popular yet verbose format, SV2 uses binary. Compared to SV1, the average message is over 50% smaller!

Pooled Connections: In SV1, the TCP connections between devices and pools are one-to-one. With SV2's new mining proxy, messages are collected and forwarded to and from the pool as one big unit, reducing bandwidth, CPU load, and infrastructure costs. This message aggregation lowers the total amount of data sent by reducing overhead and redundancy.

*at least officially, there are non standard ways to get around this

Timeline

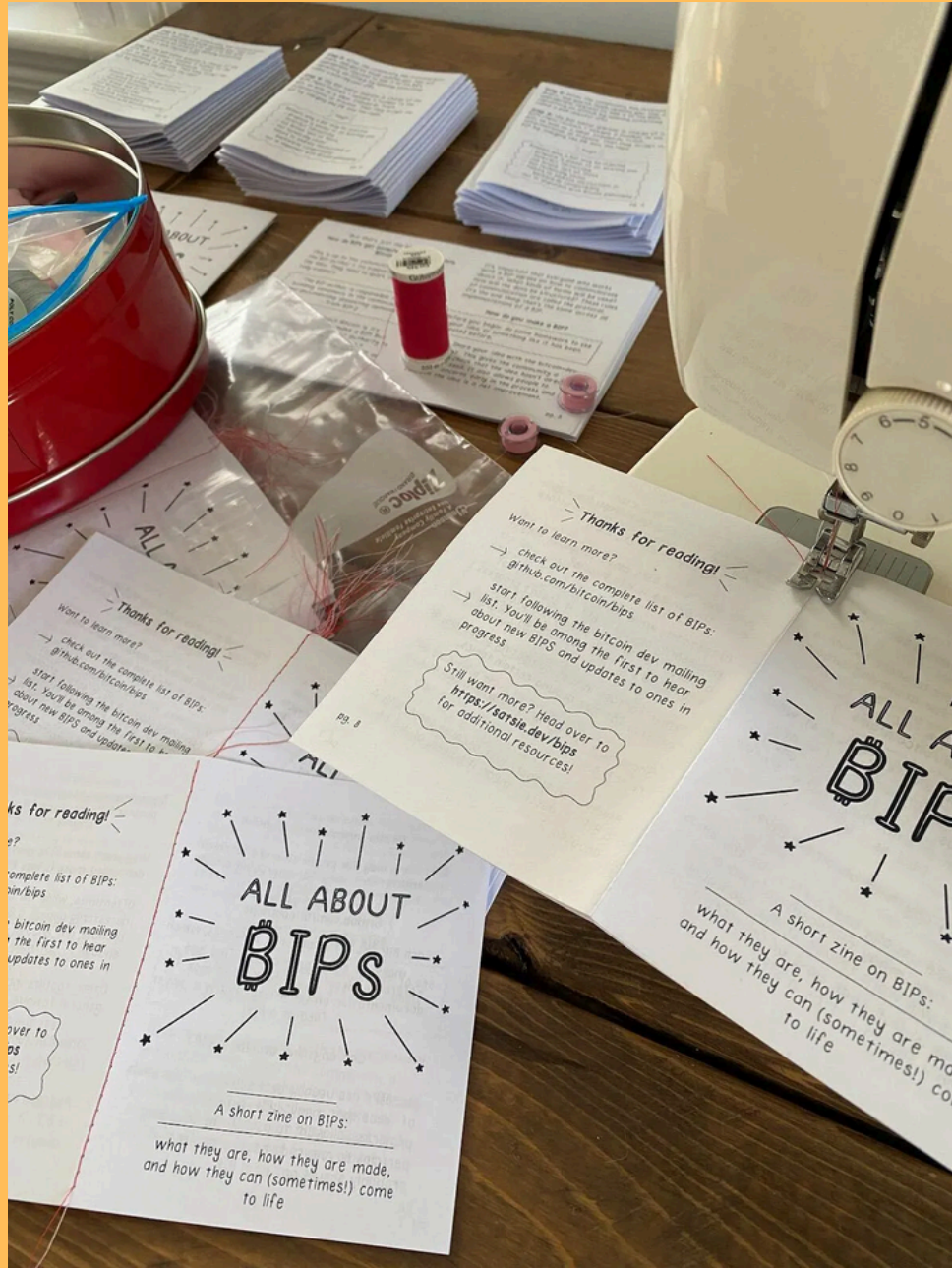
April 2020
Brains releases the first (partial) implementation of SV2.

October 2022 SRI is released. Additional releases followed April 2023 and March 2024.

@SATSIE

SATSIE.DEV/ZINES

IMAGES: HERECOMESBITCOIN.ORG

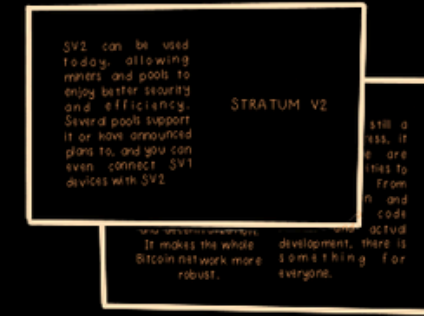


1. Print

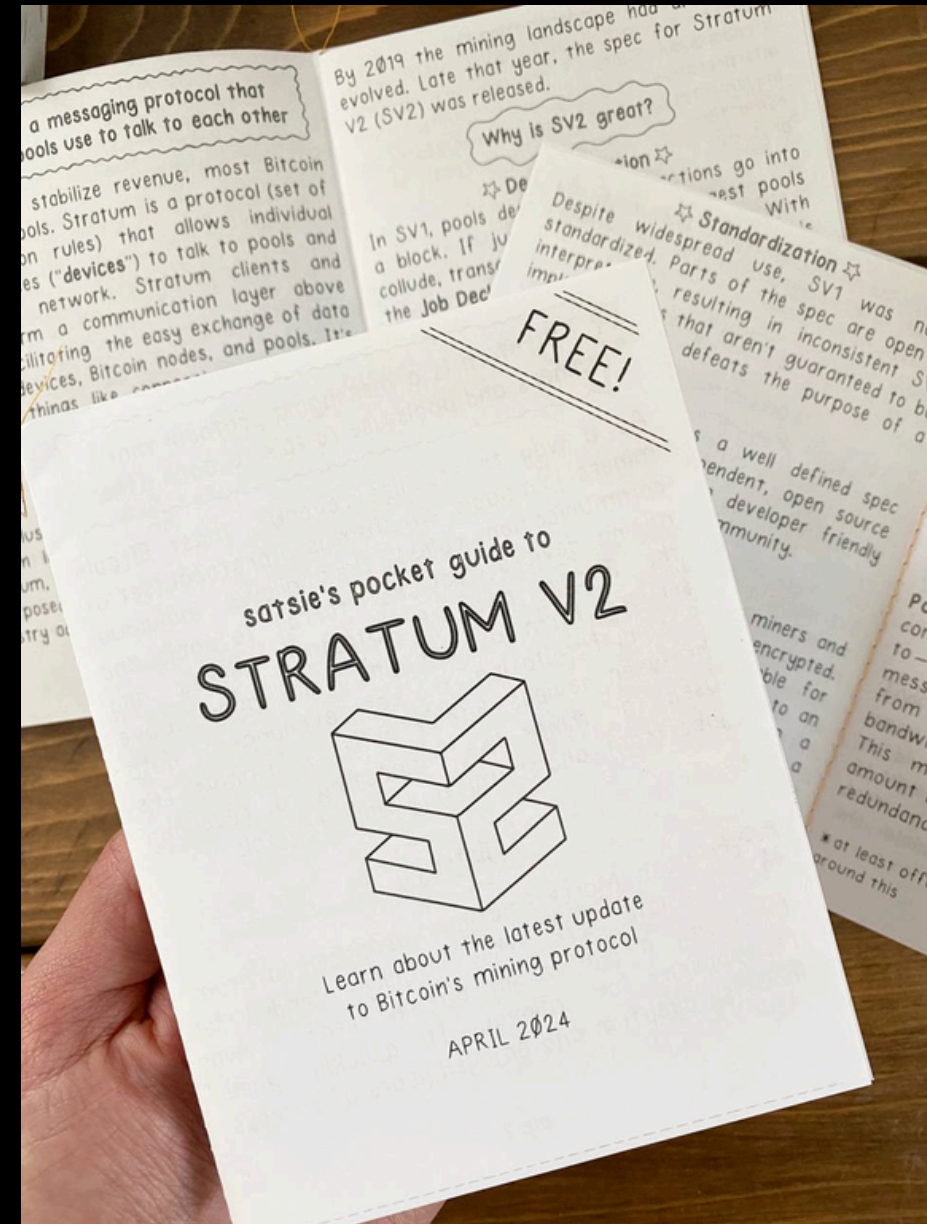


2. Cut

3. Stack

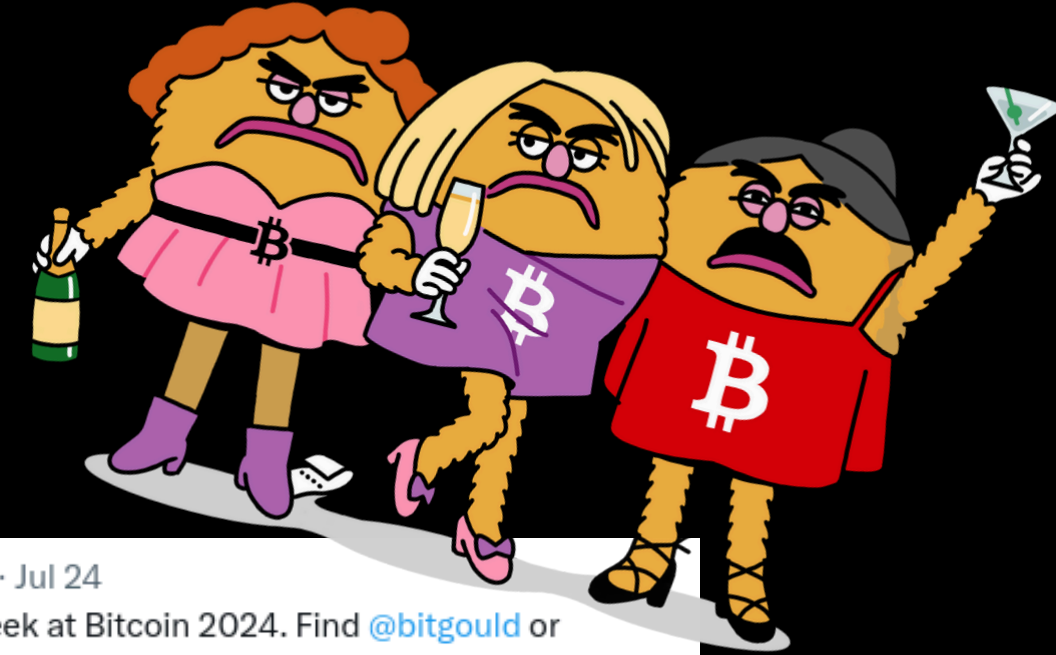


4. Fold



print
at
home

share
with
friends

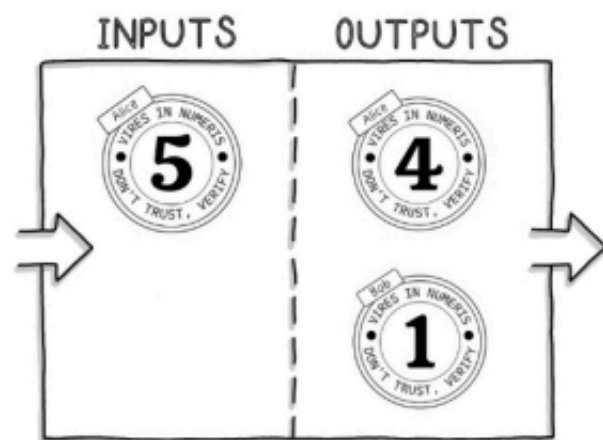


Payjoin is a technique for batching Bitcoin transactions while preserving privacy and block space.

Recall that:

1. Bitcoin uses the UTXO model, and
2. coins (transaction inputs and outputs) can be of any value

Pretend Alice has 5 BTC in her wallet and she sends 1 BTC to Bob. The transaction (tx) looks like this:



pg. 2

How BIP-77 works

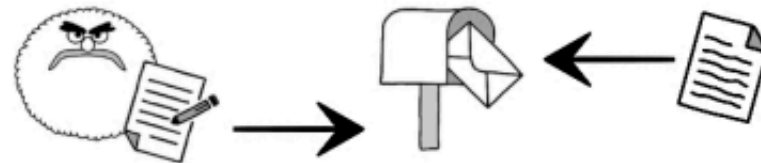


Bob: I want to start using payjoin. Can I have a mailbox?

Directory Server: Sure! Here's your address.

Bob: Hey Alice, anytime you want to send me BTC, use my mailbox so we can make it a payjoin.

Alice: Ok. I want to send you BTC. I've started a transaction, and am putting it in your mailbox. Add your input and it will be a payjoin.



Bob: Look! I have a payjoin transaction in my mailbox! Let me add my input and put it back in the mailbox.

Alice: My turn to check the mailbox. The payjoin transaction is in there and it's complete. Now I can broadcast it to the network!



pg. 7

new zine about payjoin



@SATSIE

SATSIE.DEV/ZINES

IMAGES: HERECOMESBITCOIN.ORG



satsie.dev/zines



SAVING SATOSHI

SAVINGSATOSHI.COM

A FUN, GENTLE WAY TO LEARN HOW BITCOIN WORKS

A beautiful, student focused game. A sci-fi epic.

A welcoming entry point into bitcoin tech.

Reflective of the best parts of bitcoin.



You're mining now

The code you wrote in the previous lesson to compute hashes over and over again is running.

It will stop once it's found a hash with ten leading zeroes.

See the nonce field incrementing? That's how many hashes you have tried so far!

Blocks found 0 of 100

Nonce

87.45*10¹⁶

Hashes per second

43.95*10¹⁵

100x

Running

0 Transactions confirmed

0.0000 Bitcoin earned

Enter Anything
diddums

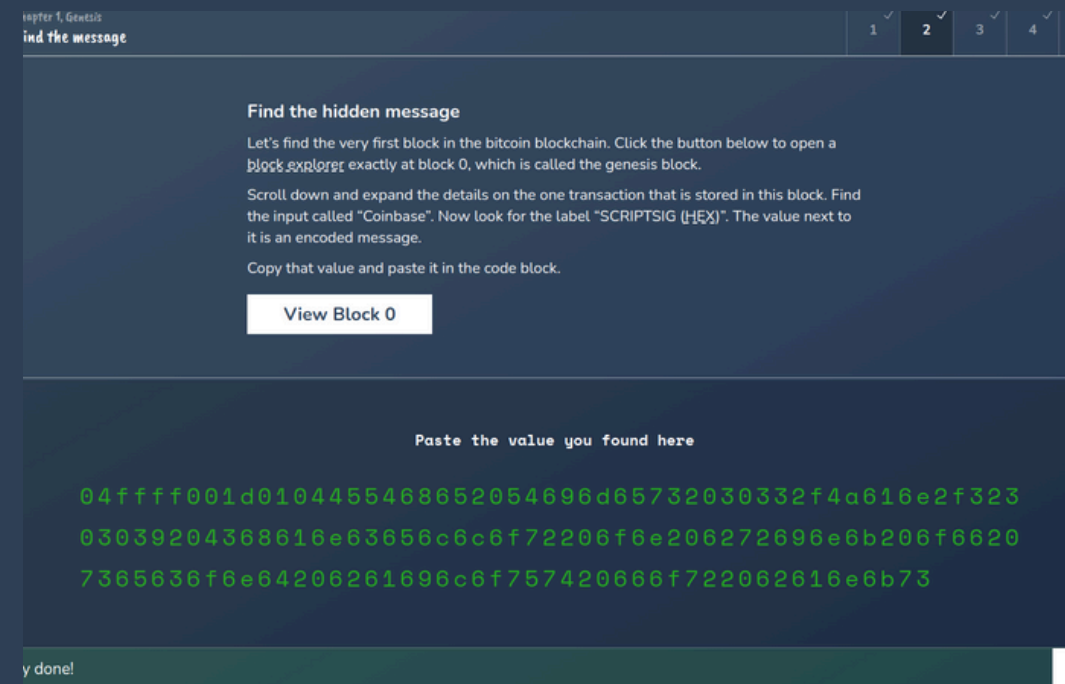
Below you will see your input converted to a hash
bb2a e32b 501f 7041 fd2e ba65 1dbd 9c00
8315 078f 3222 8cbf 81d1 8597 54c6 de78

MADE TO INSPIRE A NEW GENERATION TO FALL IN LOVE WITH BITCOIN

Understanding from a technical standpoint seals the deal.

The learning should be as cool and unique as the technology.

Complements existing resources.



FOR EDUCATORS AND THE TECHNICALLY CURIOUS, REGARDLESS OF BITCOIN BACKGROUND

Teachers and
their students

Those struggling to
progress in their
bitcoin journey

All levels

Bitcoiners and non-
Bitcoiners alike

The screenshot displays a learning interface for Bitcoin. At the top, it says "Chapter 8, Building blocks" and "Find the Smallest Transaction Block". The interface is divided into several sections:

- Block Data:** Explains that each Bitcoin full node has a database where blocks are stored and indexed by their hash. It notes that the JSON-RPC API returns block data as JSON objects with a `txs` property. A task is given: "Retrieve all the block candidates at height 6929996 and print the hash of the block with the fewest transactions in it."
- Code Editor:** Shows JavaScript code for finding the block with the fewest transactions at a specific height. The code uses the `Bitcoin` library to query the blockchain.
- Diagram:** A block tree diagram showing "Height 50" and "Height 51". A block at Height 50 (prev: 307b4d5c, hash: 5d825b7a) has two children at Height 51 (prev: 5d825b7a, hash: f06a70db and prev: 5d825b7a, hash: c1a033dd).
- Text:** A note about "Vanderpoole" mining on top of invalid blocks, and a reminder that block objects have a "prev" property.
- Code Editor (Bottom):** Shows JavaScript code for calculating transaction fees and block subsidy, and a function to validate a block.
- Image:** An illustration of an elderly man with a white beard and glasses talking on a mobile phone, and a woman with dark hair and glasses speaking into a microphone.

REAL LEARNING THAT HELPS YOU UNDERSTAND, EXPLAIN, AND USE BITCOIN BETTER

It's ambitious. You're ambitious.

Bitcoin is not inevitable. It's under attack.

Bitcoin is for everyone

You care about Bitcoin and all the things that make it special

Vanderpoole says he signed a message with Satoshi's keys:

-----BEGIN BITCOIN SIGNED MESSAGE-----

I am Vanderpoole and I have control of the private key Satoshi used to sign the first-ever Bitcoin transaction confirmed in block #170. This message is signed with the same private key.

-----BEGIN BITCOIN SIGNATURE-----

H4vQbVD0pLK7pkzPt08BHourzsBrHMB3Qf5oYVmr741pPwU2m6FaZZmxh4ScHxFoDe1FC9qG0PnAU15qMFth8k=

-----END BITCOIN SIGNATURE-----

What does this even mean?

Start

Chapter 5, Derive the message
Derive the message

1 ✓ 2 ✓ 3 ✓ 4 ✓ 5 ✓ ?

Derive the message from the transaction

It should be clear by just looking at the block explorer web page that a Bitcoin transaction has many different parts. Some parts are just small numbers and some parts are larger chunks of data. The Bitcoin protocol has a very specific algorithm for creating messages from transactions, so those messages can be signed by private keys.

We will summarize the process outlined here. It conveniently uses [this exact same transaction](#) as an example.

To begin, we need the raw bytes that make up the complete transaction. [Our block explorer](#) can help with this. Use the "hex" API endpoint and paste the entire blob of data.

Paste the transaction blob

```
0100000001c997a5e56e104102fa209c6a852dd90600a20b2d9c352423ed
ce25857fcd3704000000004847304402204e45e10932b8af514961a1d3a1
a25fdf3f4f7732e9d624c6c61548ab5fb8cd410220181522ec8eca07de48
60a4acdd12909d831cc56cbbac4622082221a8768d1d0901fffff0200
ca9a3b0000000434104ae1a02fe09c5f51b13905f07f06b99a2f7159b22
25f374cd378d71302fa28414e7aab37397f554a7df5f142c21c1b7303b8a0
0226f1baded5c72a704f7e6cd84cac0286bee000000043410411db93e1
d0db8a016b40840f8c53bc1eb68a382e97b1482e0ad7b148a6909a5cb2e0
eaddfb84ccf9744404f82e100bf09b864f9d4c03f0909b8643f856b412a3
ac00000000
```

This is the raw transaction with each component labeled:

version: 01000000
number of inputs: 01
hash of the tx that input #0 came from: c997a5e56e104102fa209c6a852dd90600a20b2d9c352423edce25857fcd3704
index of input #0 in the funding transaction: 00000000
scriptSig to authorize spending input #0: 4847304402204e45e10932b8af514961a1d3a1a25fdf3f4f7732e9d624c6c61548ab5fb8cd410220181522ec8eca07de4860a4acdd12909d831cc56cbbac4622082221a8768d1d0901
input #0 sequence: ffffffff
number of outputs: 02
output #0 value (10 BTC or 1,000,000,000 satoshis): 00ca9a3b00000000
output #0 scriptPubKey (Hal Finney's public key plus OP_CHECKSIG): 434104ae1a02fe09c5f51b13905f07f06b99a2f7159b2225f374cd378d71302fa28414e7aab37397f554a7df5f142c21c1b7303b8a00226f1baded5c72a704f7e6cd84cac
output #1 value (40 BTC or 4,000,000,000 satoshis): 00286bee00000000
output #1 scriptPubKey (Satoshi's own public key again, for change): 43410411db93e1d0db8a016b40840f8c53bc1eb68a382e97b1482e0ad7b148a6909a5cb2e0eaddfb84ccf9744404f82e100bf09b864f9d4c03f0909b8643f856b412a3ac
locktime: 00000000



An expression of our love for Bitcoin

Our contribution towards making Bitcoin the best it can be

Scratch your own itch

Chapter 7, Building blocks
Assemble a block

You can make the following assumptions to complete your mission:

- All transactions in the mempool have already been verified as valid.
- The coinbase transaction, and the weight it contributes to the block, can be ignored.

You can view the entire raw mempool JSON file [here](#).

Or browse an excerpt of the file in this table for some basic patterns:

Transaction ID	Fee Satoshis	Weight Weight units (WU)	Ancestors
b27f86d3	43430	2020	
c27b4d2e	30168	1676	bd1d83ca f29aec75
18725711	5520	1840	3c64a457 3c8abf73
92b1ecf5	24302	1676	398695a6 10025d80
8e8c8624	8990	1160	01f6094b
5f6c9a80	13716	1524	64121ab1
e140fa46	13020	1488	7675c31c
d7066e71	16416	1152	
88016f17	15200	1600	5e518bbe
8fa820d5	20221	1108	38a62dcc

[View the json file here](#) for the rest of the transaction data

```
JavaScript Python
1 const json = require('./mempool.json');
2
3 class MempoolTransaction {
4   constructor(json) {
5     this.txid = json.txid;
6     this.weight = json.weight;
7     this.fee = json.fee;
8     this.parents = json.parents;
9   }
10 }
11
12 // Fix this!
13 function assembleBlock(mempool) {
14   const block = [];
15   for (const tx of mempool) {
16     block.push(tx.txid);
17   }
18   return block;
19 }
20
21 function importMempoolFromJson(json) {
22   const mempool = [];
23   for (const tx of json) {
24     mempool.push(new MempoolTransaction(tx));
25   }
26   return mempool;
27 }
28
29 function run() {
30   const mempool = importMempoolFromJson(json);
31   const block = assembleBlock(mempool);
32   return block;
33 }
34
35 script output
36 waiting for you to run the script...
```

Run the script



A LABOR OF LOVE

SAVINGSATOSHI.COM