Bits on Bitcoin

Justin Moore
March 29, 2019 | Insight Decentralized Consensus

Want to know one of your blind spots?

The hardware you are using is not secure.

… neither is the software.

Assuming hardware is out of our control, can we do any better in software?

I believe we can.

And I have an idea for it.

Hardware requires software.

We generally call this specialized software firmware.

When downloading firmware, how does one establish trust using the Internet?

Digital signatures and checksums -- both require user interaction.
Let’s talk checksums.

Verifying integrity (Developer)

1. Validates their code before uploading by computing a checksum
2. Publishes code to a web-server, displaying the checksum next to the download link

Verifying Integrity (User)

1. Note the checksum when you download the file
2. Compute the checksum once downloaded
3. Compare checksum values

To be clear - this is meant to protect against data corruption, not prevent a man-in-the-middle attack.

E.g.

Via Direct Download

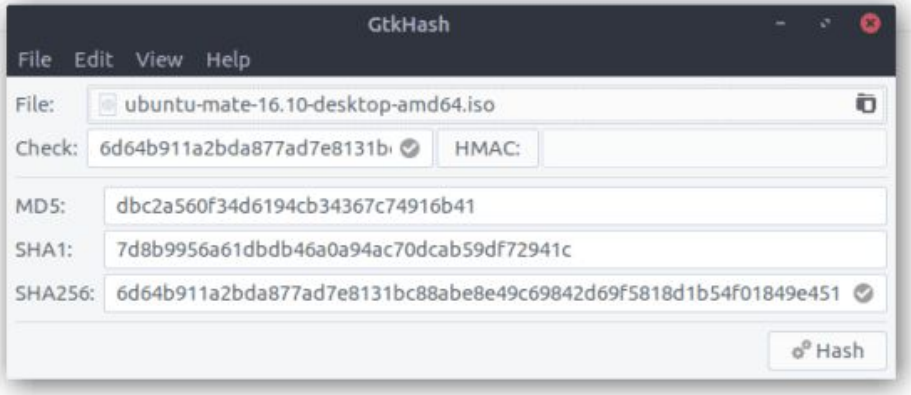
If preferred, you can also download the images over HTTP.

ubuntu-mate-16.10-desktop-amd64.iso

Download Size: 1.7 GB

SHA256 Checksum: 6d64b911a2bda877ad7e8131bc88abe8e49c69842d69f5818d1b54f01849e451

How to verify downloads



Can we use Bitcoin beyond money?

We could potentially embed information into tiny transactions

This would look like normal transactions, but contain our data

By weaving these transactions together to recover the code

My first idea was to use the value field as I noticed that a Satoshi maps to a bit

The smallest unit of bitcoin is a Satoshi (0.00000001 bitcoin)

If you ignore the “0.” this looks like a byte (8-bits in computing).

E.g. 11111111 satoshis looks like 0xFF satoshis in hex

Reminder: hex is base 16, so 0xF=1111, 0xE=1110, 0xD=1101, etc.

But thinking this through … it would quickly get too expensive.

Next I thought about encoding/compressing in the value field

Since satoshis are in decimal, not binary, maybe we could put more info in the value field.

I played around with this some this week but it still seems like it's not an optimal way to go.

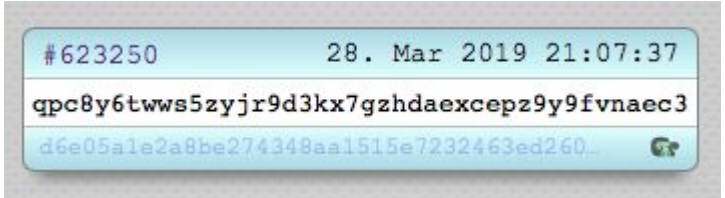
Landed on: let’s look to what other folks do

There seem to be a couple methods to accomplish writing data to the chain and I’ve found an example I used as a launchpad, but want to look into the source in more detail and then build out my own version.

demo time

1) Write the starting address on chain

qpc8y6twws5zyjr9d3kx7gzhdaexcepz9y9fvnaec3
^ cash format of 1BFZjxdWM1gDukNGFMQWmKUv12Lto4mun9



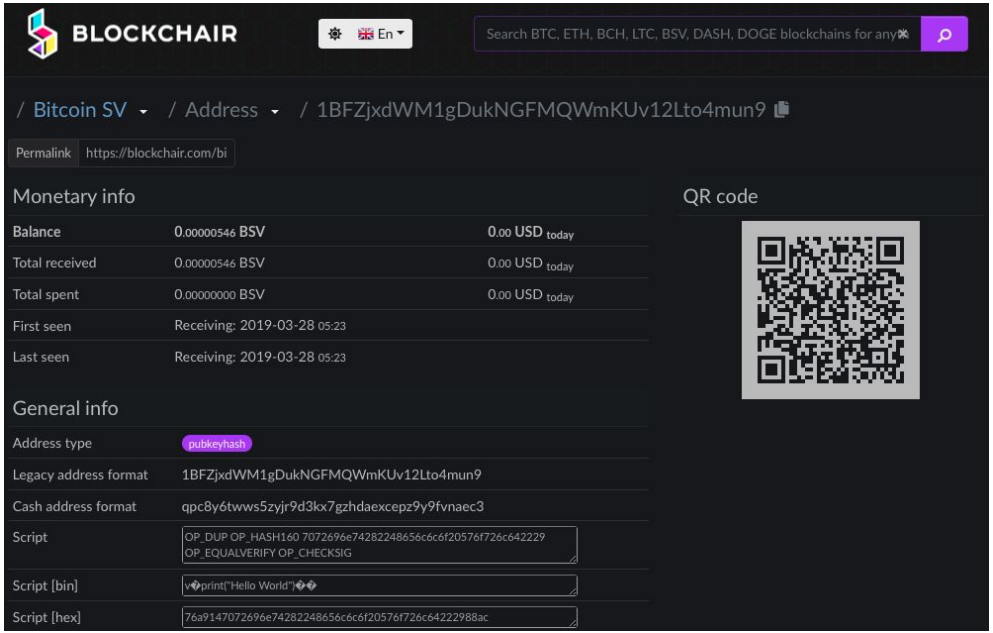
2) Hit this address in a block explorer (for now)

https://blockchair.com/bitcoin-sv/address/qpc8y6twws5zyjr9d3kx7gzhdaexcepz9y9fvnaec3

same as

https://blockchair.com/bitcoin-sv/address/1BFZjxdWM1gDukNGFMQWmKUv12Lto4mun9

2) Hit this address in a block explorer (for now)



2) Hit this address in a block explorer (for now)

Legacy address format	1BFZjxdWM1gDukNGFMQWmKUv12Lto4mun9
Cash address format	qpc8y6twws5zyjr9d3kx7gzhdaexcepz9y9fvnaec3
Script	OP_DUP OP_HASH160 7072696e74282248656c6c6f20576f726c642229 OP_EQUALVERIFY OP_CHECKSIG
Script [bin]	v\u2026print("Hello World")\u2026\u2026

3) Run the code I wrote on chain previously

print(“Hello World”)

> Hello World

Thoughts on the future

Miners or specialized nodes will provide this kind of weaving service (searching the chain and remembering where things live).

You could imagine the possibility of an on-chain PageRank for a micro fee instead of giving up a digital replication of your self to be packaged and sold to advertisers.