Bitclamp

A Permanent and Anonymous Publishing Platform Over Bitcoin

Joe Testa
jtesta@positronsecurity.com
Twitter: therealjoetesta

Rochester Security Summit 2016
October 5, 2016

Quick Primer on Bitcoin

- Bitcoin is a 100% digital currency.
  - Protects against fraud using strong cryptography.
- It is decentralized.
- Payments can be anonymous.
  - Emphasis on *can be*...
- Payments are *permanent*.

Why Use Bitcoin To Publish?

- Anything that goes into the Bitcoin blockchain is *permanent*.
  - Its also (semi) anonymous.

- Removing anything from it implies you have the ability to revert transactions for *everyone*.
  - You'd need to hash faster than 1.9 EH/s.

- Perfect situation for whistleblower-type documents.

Outline

- I'm going to cover:
  - Some bitcoin technicals & background
  - Two prior methods of publishing; analyze their shortcomings
  - Unveil a new method; analyze its properties
  - Discuss some advanced features

Bitcoin Background

- Bitcoin has its own scripting language to control transactions (!).

- The stack-based language supports:
  - Conditional statements
  - Bitwise logic
  - Arithmetic
  - Cryptographic functions

- Docs: https://en.bitcoin.it/wiki/Script

Bitcoin Background: Transaction Fees

- Miners want money.
  - The block reward is the main motivator.
  - Transactions can have fees attached which also go to the miner.

- Fees aren't strictly mandatory, but zero-fee transactions can be ignored for awhile.
  - Or not, depending on what other transactions are currently doing.
  - It functions as a free market.
  - 0.0002 BTC per KB ($0.12 per KB)

Prior Publication Methods

- Bitcoin addresses are encoded ECDSA keys:

Base58(0x00 + RIPEMD-160(SHA256(pubkey)) + checksum)

= 1BvBMSEYstWetqTFn5Au4m4GFg7xJaNVN2

- Base58 is Base64 with some characters removed.
  - It is also reversable.

Prior Publication Methods

- The RIPEMD-160 output is 20 bytes.
- Take a 20 byte chunk of data, and apply Base58 to make an address.
- Send a small amount of BTC to that address.
  - Since a transaction can have 256 outputs, you can put in 20 x 256 = 5K bytes.
  - Use another transaction to continue the next block.

Prior Publication Methods: Example

Example for publishing “ABCDEFG...”

1. Convert to ASCII bytes (0x65, 0x66, 0x67, …)
2. Calculate checksum.
3. Create Bitcoin address:

   Base58(0x00 + ASCII bytes + checksum)

   = 1j3AzhncjRAxZQjiFikyBkCJwFVKNWTKVx
4. Send small amount to this address.
5. Repeat.

Prior Publication Methods

- Someone published the PDF of the original Bitcoin paper using this method.
- I can't find where it is anymore.
- This says a lot about how useful this method is.

Prior Publication Methods

- This method has two major disadvantages:
  - You're destroying coins.
  - Nobody has the private keys to the “public keys” you're sending to.
  - UTXO pollution

  - Its expensive.
  - 0.00000504 BTC (?) per 20 bytes.
  - AND have to pay transaction fees!

- 1 MB would cost *at least* 0.47 BTC ($285 USD)
  - Actual cost could be 0.71 BTC ($430 USD)

Prior Publication Methods

- Bitcoin devs didn't like the UTXO pollution.

- A compromise was reached: the network would allow the following:

  OP_RETURN <40, 80 bytes>

- *OP_RETURN* results in failure, making the transaction provably unspendable.
  - Can be pruned from UTXO database.

Prior Publication Methods

- This method is very limiting because:
  - Payload is tiny.
  - Hard to reference other transactions.

  - Only one OP_RETURN per transaction.
  - Can't put one in each of 256 outputs.

- People currently use this to publish hashes, silly messages: http://eternitywall.it/

Goals For New Method

- I want a way to publish data of arbitrary length.
- I want to be able to search and/or pull down all data.
- I want it to cost as little as possible.
- I want tools available to everyone.

Bitcoin Background: Standard Transactions

- Transactions are evaluated by *IsStandardTx()* method call.
  - If it fails this check, it will not be relayed around the network!
- Many opcodes in the scripting language are no longer allowed.
- I had to find a way to beat this.

New Method: Multi-sig Addresses

- Bitcoin supports “multi-sig” transactions.
  - *n* of *m* keys need to provide signatures to spend coins, n <= *m* <= 15.
  - Could require any 3 of 7 keys, any 5 of 11 keys, etc.
- Useful to protect against malware/theft.
  - In 2 of 2 scheme, one key compromise is OK.

New Method: Multi-sig Addresses

- Since up to 15 keys are supported, lets use them!
- We generate one legit key, then say 1 out of these 15 are needed to spend.
- The other 14 keys are really data.
  - Raw ECDSA keys are used, which are 32 bytes.
- Hence, 14 x 32 = 448 bytes!

New Method: Multi-sig Addresses

- Advantages:
  - Large payload
  - Transactions remain spendable
  - 448 bytes *per output*
  - Only need to pay transaction fees (if at all)

- Transactions are limited to 10,000 bytes however (including all headers, etc) (?).

- Can get 67% efficiency.

New Method: Multi-sig Addresses

- Publishing 1 MB will cost *up to* 0.31 BTC ($188 USD)
  - Contrast with maybe $430 for first method.
- Could cost much less if you're patient.
- Publication time: 14 hours.
  - Tradeoff between fees paid and time to publish.
  - Very low fee can result in weeks or months.

New Method: Multi-sig Addresses

- Costs can be cut down to much less than $188 per 1MB.
- Compression algorithms are automatically applied.
  - Plain text, source code compresses extremely well.
- Do you really need 99,000 megapixel photos?
  - No. No you do not.
- Note that transaction fees depend on time of day.

Other Blockchains Are Viable

Other Blockchains: Dogecoin

- Advantages:
  - Fast block times: 1 minute (vs 10 minutes)
  - Extremely low/no transaction fees

- Disadvantages:
  - Much lower hash rate
  - Will it survive?
  - Publication may not be permanent
    - But maybe that's ok...

Other Blockchains: Dogecoin

- 1 MB can be published for $0.43.
- 1 MB can be published in 2 hours.
- Since the blocks are mostly empty, multiple transactions can be sent simultaneously.
  - I'm working on code that can push that down to… 15 minutes?

Special Features: Temporal Encryption

- By default, all publications are encrypted with a random key.
  - Uses GPG2
- In the last block, the key is divulged.
  - Protects the content while its being published.
- Encryption can be disabled.
  - In some cases, this may be better.

Special Features: Deadman Switch

- As an insurance policy, an encrypted archive can be published without the key.
- The user sets up a process to check in every X hours, days, etc.
- If the user is arrested or killed, the key is automatically published!

Can Bitcoin Block This?

- Not without interfering with existing functionality.
  - Multi-sig support can be reduced from 15 keys to something lower.
  - Can just bump up the number of outputs per TX.

- They might not even care.
  - The UTXO database isn't polluted.
  - Miners still get transaction fees.

Code & Web Front-End

- The code is available **now**: https://github.com/jtesta/bitclamp

- Its currently in beta.
  - Will reach stable status by November 2016.

- There will be a website front-end for it.
  - You'll be able to publish with it easily.
  - Browse, search, and download content.
  - ETA: winter 2016

Questions?

Code: https://github.com/jtesta/bitclamp

Joe Testa

jtesta@positronsecurity.com
Twitter: therealjoetesta