Forks and Upgrade Mechanisms

Or: what the hell is Bitcoin?!!

Agenda

- Hard Forks
- Soft Forks
  - Examples
  - Unexpected power of soft forks
- Comparison
- A brief history of deployment methods
- Outlook

Bitcoin Layers

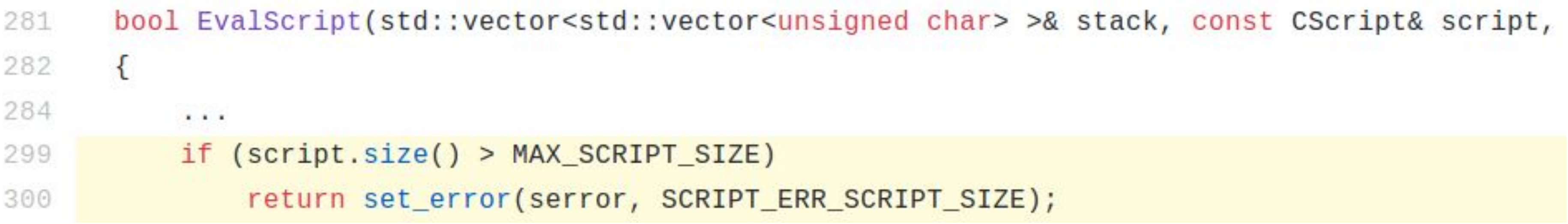
1. **Consensus**
2. Peer Services
3. API/RPC
4. Applications

(defined in BIP123 - BIP Classification)

Consensus Rules

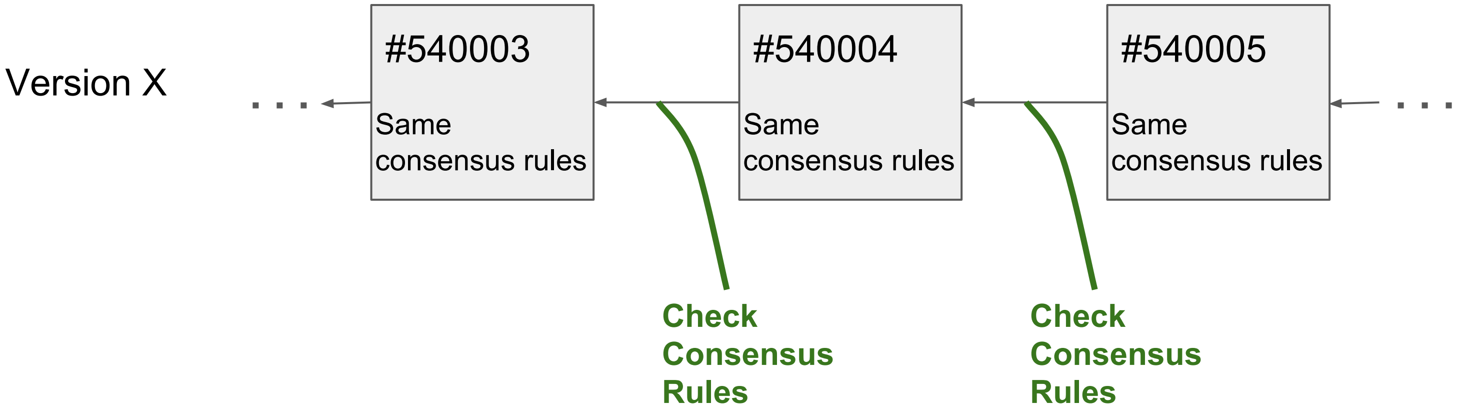
- Define what constitutes a valid block
  - Formatting
  - Proof of work, coin supply, ...
  - Script execution, signature checks
  - Double spend check
  - Various limits (weight, script sizes, ...)
  - Etc.
- Most difficult to change
- Broadly two ways of doing so:
  - via **Hard Forks**
  - via **Soft Forks**

Consensus Rules

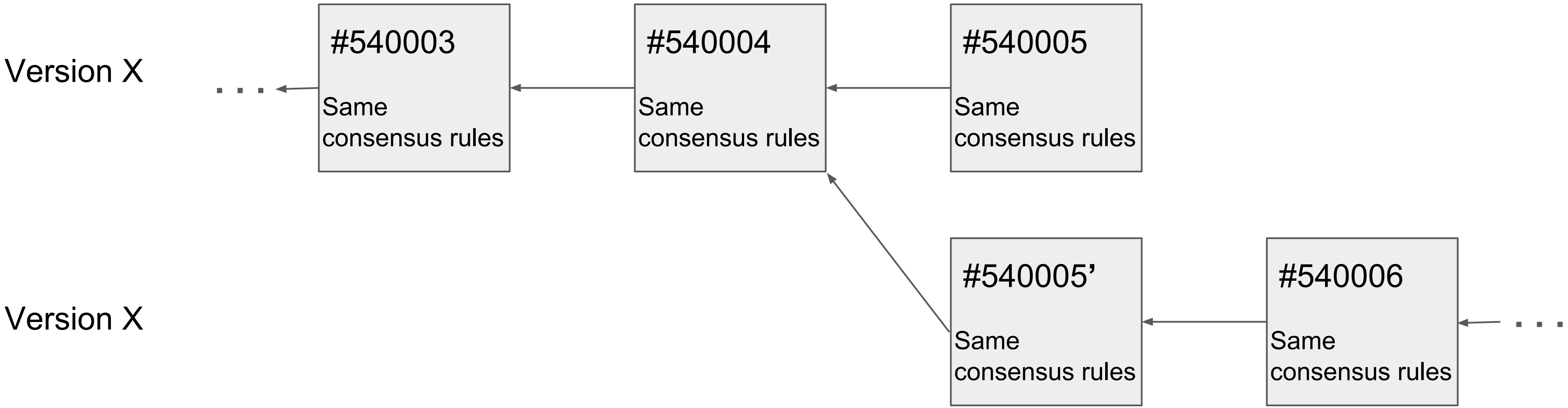


```
bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script,
{
    ...
    if (script.size() > MAX_SCRIPT_SIZE)
        return set_error(serror, SCRIPT_ERR_SCRIPT_SIZE);
```

Normal Operation



A fork, but not an upgrade: orphaned blocks



Version X

#540003

Same consensus rules

#540004

Same consensus rules

#540005

Same consensus rules

Version X

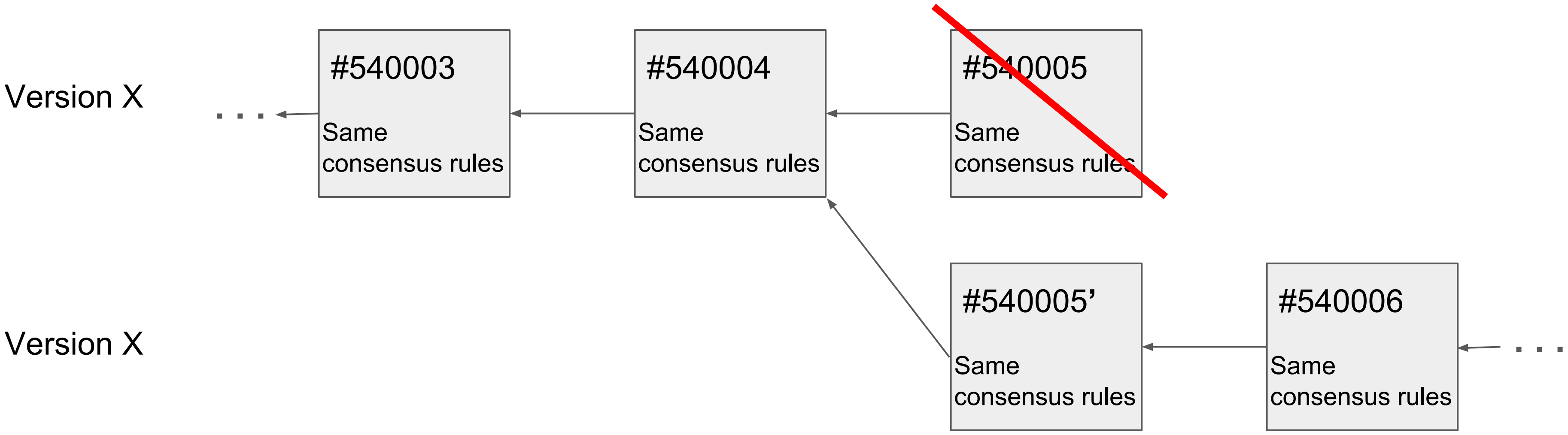
#540005’

Same consensus rules

#540006

Same consensus rules

A fork, but not an upgrade: orphaned blocks



Longest/heaviest chain (*under same consensus rules)* is the main chain. Shorter forks are abandoned / orphaned.

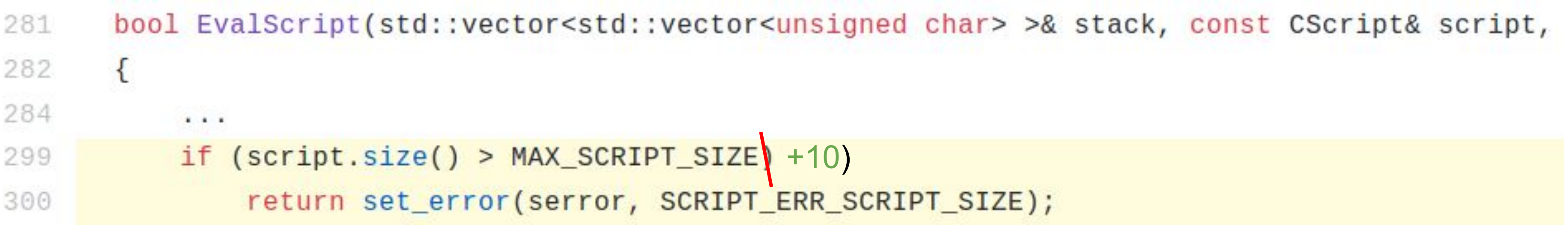
Hardforks

- Hardforks remove or relax consensus rules
- Blocks following new rules are **rejected** by old nodes
  - Not forwards compatible
- Can change pretty much anything about the coin
- Backwards compatible? Usually yes, but not necessarily!

Examples:

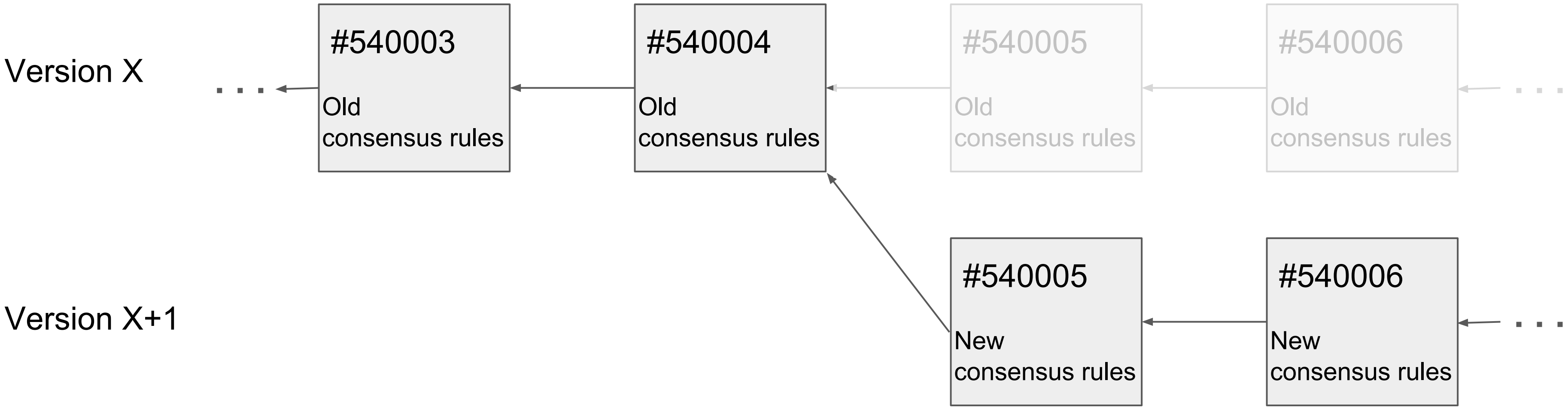
- Change Proof of work function
- Change block header magic prefix to `21lectures`
- Add new OP codes to script
- Increase coin supply, script size limit, …

Hardforks

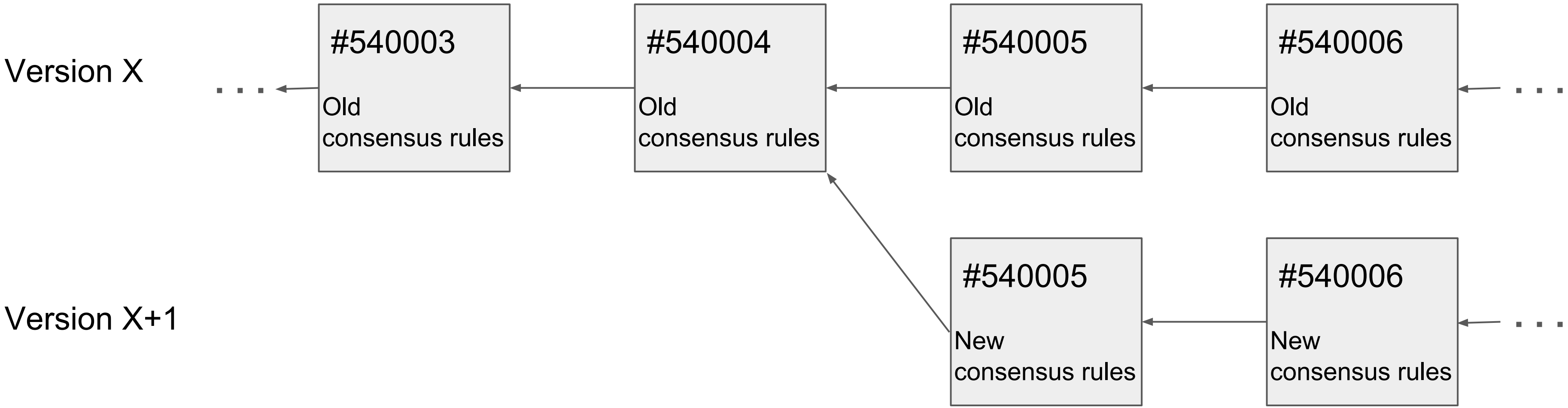


```
bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script,
{
    ...
    if (script.size() > MAX_SCRIPT_SIZE~~)~~ +10)
        return set_error(serror, SCRIPT_ERR_SCRIPT_SIZE);
```

Hardforks


Version X
#540003
Old
consensus rules
#540004
Old
consensus rules
#540005
Old
consensus rules
#540006
Old
consensus rules
Version X+1
#540005
New
consensus rules
#540006
New
consensus rules

Hardforks


Version X

#540003
Old
consensus rules

#540004
Old
consensus rules

#540005
Old
consensus rules

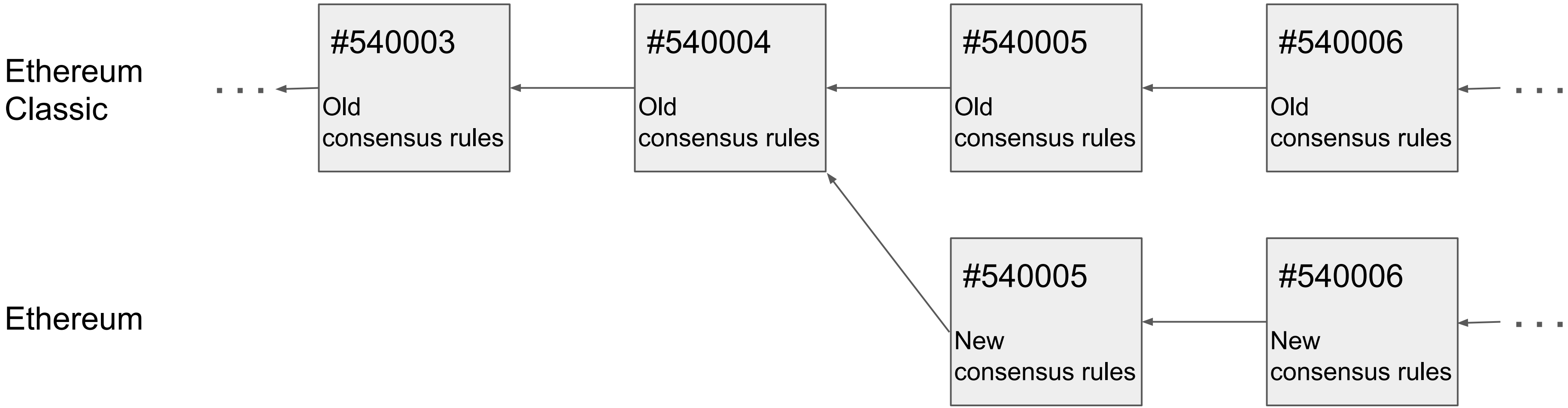
#540006
Old
consensus rules

Version X+1

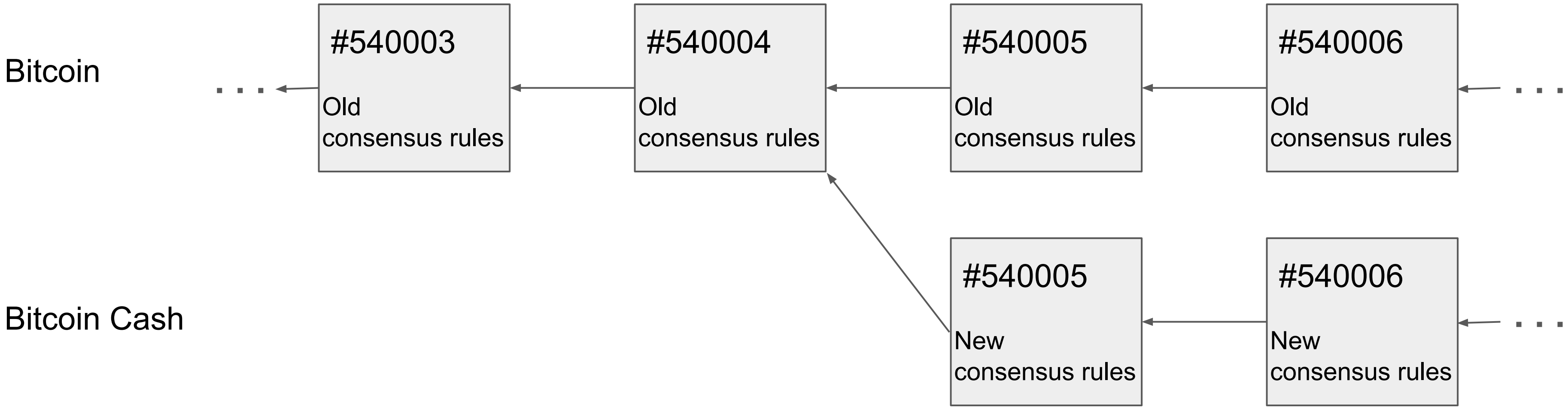
#540005
New
consensus rules

#540006
New
consensus rules

Hardforks


Ethereum Classic
#540003
Old consensus rules
#540004
Old consensus rules
#540005
Old consensus rules
#540006
Old consensus rules
Ethereum
#540005
New consensus rules
#540006
New consensus rules

Hardforks


Bitcoin
#540003
Old
consensus rules
#540004
Old
consensus rules
#540005
Old
consensus rules
#540006
Old
consensus rules
Bitcoin Cash
#540005
New
consensus rules
#540006
New
consensus rules

Hardforks - Complications

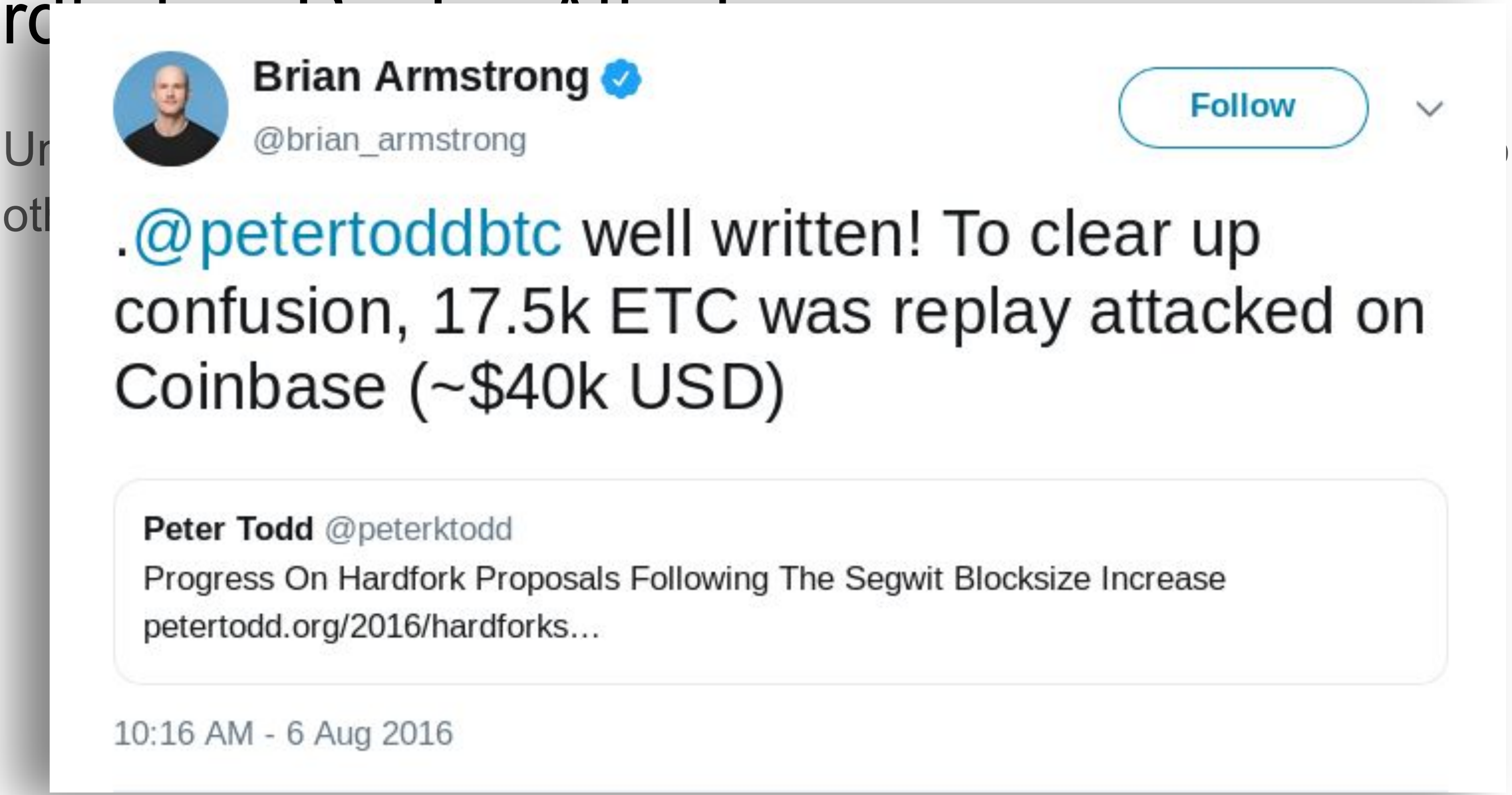
- In a contentious upgrade, the chain can permanently split into two chains
- Before the actual fork, no one knows for sure if one side will be dominant, or if there will be two chains
- Which inherits the name?
- Which version to support as an exchange, wallet maker, user, …?
- A lot of uncertainty and overhead.

Hardforks - Replays

- Unless replay-protected, transactions from one chain can be replayed on the other chain

Hard

- Ur n the
  otl



Brian Armstrong
@brian_armstrong

.@petertoddbtc well written! To clear up confusion, 17.5k ETC was replay attacked on Coinbase (~$40k USD)

Peter Todd @peterktodd
Progress On Hardfork Proposals Following The Segwit Blocksize Increase
petertodd.org/2016/hardforks…

10:16 AM - 6 Aug 2016

Hardforks - Replays

- Unless replay-protected, transactions from one chain can be replayed on the other chain

Hardforks - Replays

- Unless replay-protected, transactions from one chain can be replayed on the other chain
- Can be fixed by making transactions be invalid on the other chain
    - Different signature hash algorithm, SIGHASH_FORKID

Hardforks - Replays

- Unless replay-protected, transactions from one chain can be replayed on the other chain
- Can be fixed by making transactions be invalid on the other chain
    - Different signature hash algorithm, SIGHASH_FORKID
- Hardforks should always come with built-in replay protection

Hardfork Deployment

- Define activation block number far enough in the future
  - Give ample time for everyone to learn about it
  - Give ample time for the ecosystem to upgrade - wallets, exchanges, infrastructure, etc.

- Code conceptually straight-forward:

```
if (blockNumber > 600000) { // expected ~ end of 2019
    // apply new consensus rules, ideally containing replay protection
} else {
    // apply old consensus rules
}
```

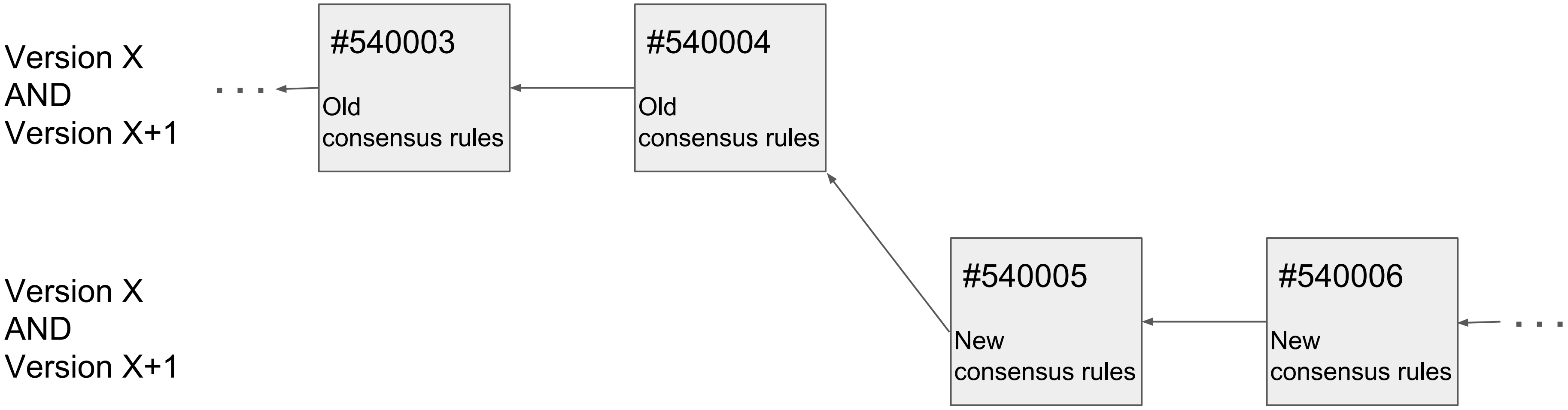
Softforks

- Softforks add or tighten consensus rules
- Blocks following new rules are **accepted** by old nodes
    - forwards compatible
- **Majority of mining power needs to upgrade. Normal full nodes do not.**

Examples:

- Decrease coin supply
- Decrease limits, e.g. script size limit
- Remove OP codes from script, or redefine an OP_NOP code

Softforks



Version X
AND
Version X+1

#540003

Old
consensus rules

#540004

Old
consensus rules

Version X
AND
Version X+1

#540005

New
consensus rules

#540006

New
consensus rules

Softforks - Examples

-

Softforks - OP_CHECKLOCKTIMEVERIFY

- Defined in BIP65
- Prevents spending an output until a certain time/block

- Introduction by modifying Script, the Bitcoin script language.
- Redefined OP_NOP2 to fail if the specified time > nLockTime

Upgraded nodes see: <time> CHECKLOCKTIMEVERIFY DROP

Non-upgraded nodes see: <time> OP_NOP2 DROP

⟶ Forwards compatible!

Softforks - Segwit (BIP141)

P2PKH

```
Input Script
<signature>
<pubKey>
Output Script
OP_DUP
OP_HASH160
<pubKeyHash>
OP_EQUALVERIFY
OP_CHECKSIG
```

Softforks - Segwit (BIP141)

P2PKH

Input Script

```
<signature>
<pubKey>
```

Output Script

```
OP_DUP
OP_HASH160
<pubKeyHash>
OP_EQUALVERIFY
OP_CHECKSIG
```

→

P2WPKH

Witness

```
<signature>
<pubKey>
```

Input Script: *empty*

Output Script

```
<0>
<pubKeyHash>
```

Softforks - Segwit (BIP141)

P2PKH

```
Input Script
<signature>
<pubKey>
Output Script
OP_DUP
OP_HASH160
<pubKeyHash>
OP_EQUALVERIFY
OP_CHECKSIG
```

→

P2WPKH

```
Witness
<signature>
<pubKey>
Input Script: empty
Output Script
<0>
<pubKeyHash>
```

→

Old nodes see

```
Input Script: empty
Output Script
<0>
<pubKeyHash>
```

Segwit Script Versioning

Witness

<signature><script>

Input Script: *empty*

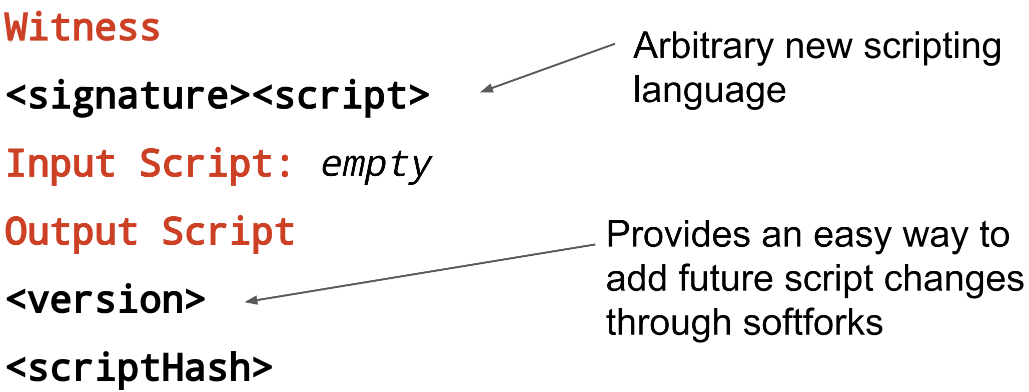
Output Script

<version>

<scriptHash>

Provides an easy way to add future script changes through softforks

Segwit Script Versioning



Witness

`<signature><script>`

Input Script: *empty*

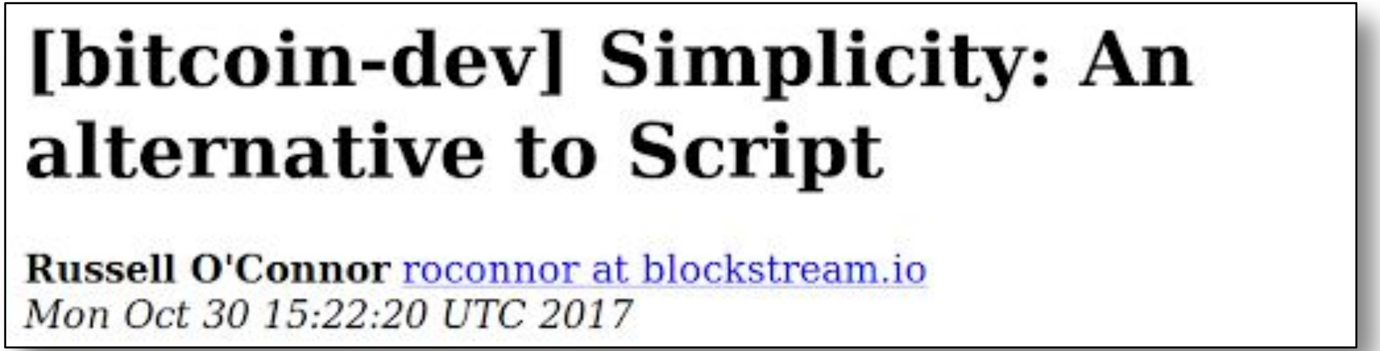
Output Script

`<version>`

`<scriptHash>`

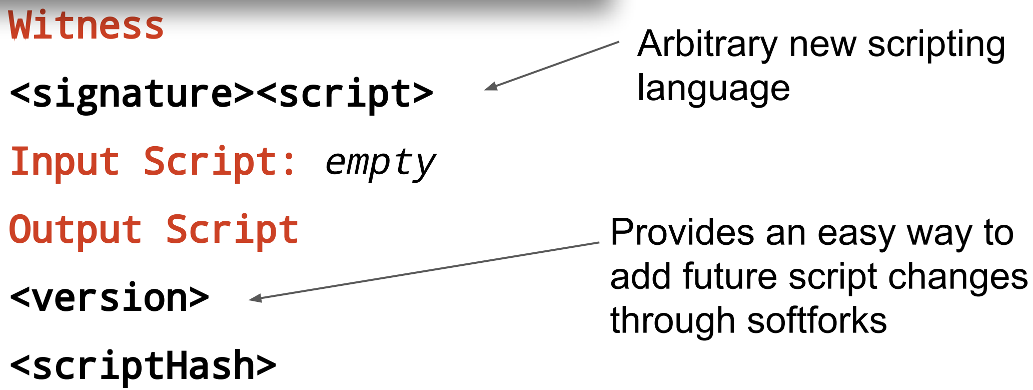
Arbitrary new scripting language

Provides an easy way to add future script changes through softforks



[bitcoin-dev] Simplicity: An alternative to Script

Russell O'Connor roconnor at blockstream.io
Mon Oct 30 15:22:20 UTC 2017



Witness

`<signature><script>`

Arbitrary new scripting language

Input Script: *empty*

Output Script

`<version>`

`<scriptHash>`

Provides an easy way to add future script changes through softforks

Unexpected Power of Softforks

- After segwit, easy to softfork in arbitrary new scripting languages

- Can softfork in any change, really!
  (commit to a parallel block with arbitrary rules in the coinbase tx, possibly prohibit any other transactions in the main block)

  Called **evil fork** or **forced soft fork** (Peter Todd)

Hardfork vs. Softfork

Hardfork

- Loosens rules
- Risk of chain-split - you have to choose if you want a specific upgrade or not
- Only one upgrade at a time.
- Previous upgrades mandatory.
- Easy to estimate support.

Softfork

- Tightens rules
- Single chain, can seamlessly handle either preference
- Multiple upgrades at a time.
- Previous upgrades not mandatory.
- Very hard to estimate support beforehand. Once activated, it is there to stay forever, even if there is barely any use.

Both:

- Can modify the protocol arbitrarily
- Opt-out by default, can opt-in by upgrading

Deploying a Softfork

- Goal: coordinate activation so that miners and users activate at the same time
- Rich history with different methods.

Deploying a Softfork

- Goal: coordinate activation so that miners and users activate at the same time
- Rich history with different methods.
- BIP16 (p2sh): flag date & 55% mining power during an activation window. Many issues.

Deploying a Softfork

- Goal: coordinate activation so that miners and users activate at the same time
- Rich history with different methods.
- BIP16 (p2sh): flag date & 55% mining power during an activation window. Many issues.
- Later bips: activate after 95% of miners signal readiness, using block versions 2, 3, …
    - only one soft fork at a time.
    - If a fork failed, there could be no more.
    - Called **MASF** - Miner Activated Soft Fork

Deploying a Softfork

- Goal: coordinate activation so that miners and users activate at the same time
- Rich history with different methods.
- BIP16 (p2sh): flag date & 55% mining power during an activation window. Many issues.
- Later bips: activate after 95% of miners signal readiness, using block versions 2, 3, …
    - only one soft fork at a time.
    - If a fork failed, there could be no more.
    - Called **MASF** - Miner Activated Soft Fork
- Then started to use block version bits to encode multiple soft forks and their state, still with 95% miner signaling for coordination (BIP9)

Deploying a Softfork

Then, segwit was slated for activation, and everything changed 😱

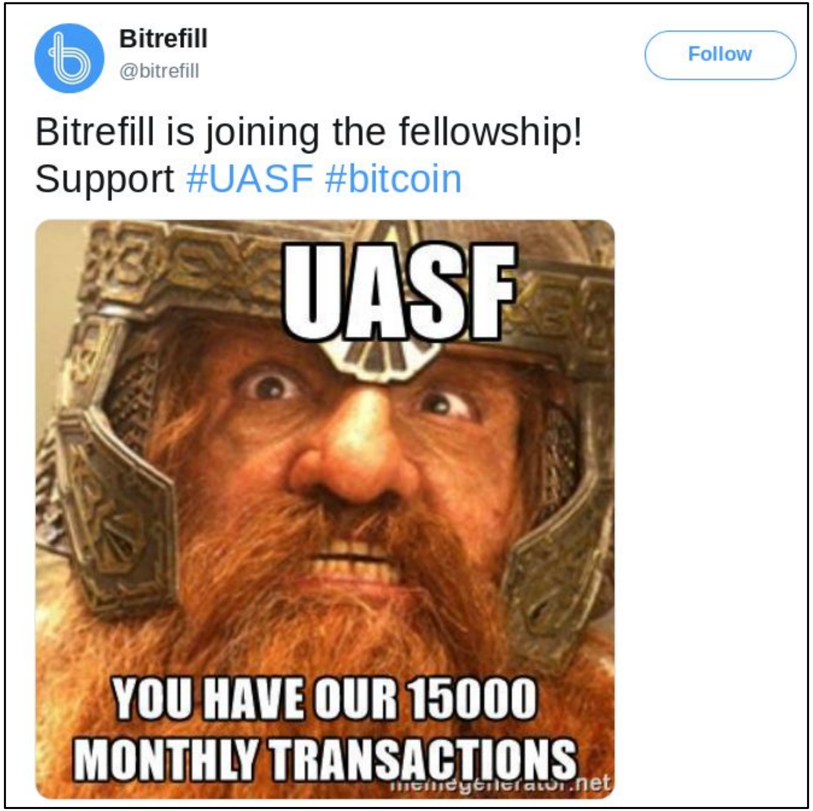
Blocksize debate, HF vs. SF, unhappy miners and Segwit2X

UASF - User Activated Soft Fork

95% of mining supermajority: signaling or voting?

Idea: if the economic majority enforced a softfork, miners would have to follow. Otherwise their blocks would be rejected and be worthless.

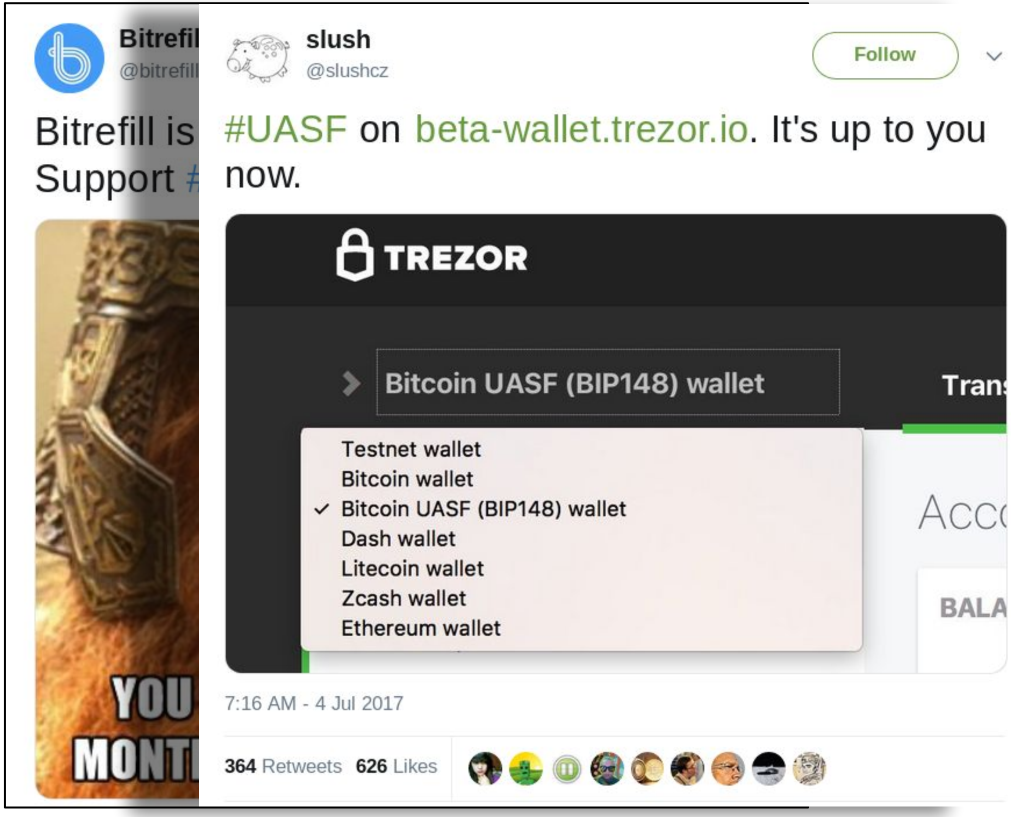
UASF - User Activated Soft Fork



Bitrefill
@bitrefill

Bitrefill is joining the fellowship!
Support #UASF #bitcoin

UASF - User Activated Soft Fork



Bitrefil
@bitrefill

Bitrefill is
Support #

slush
@slushcz

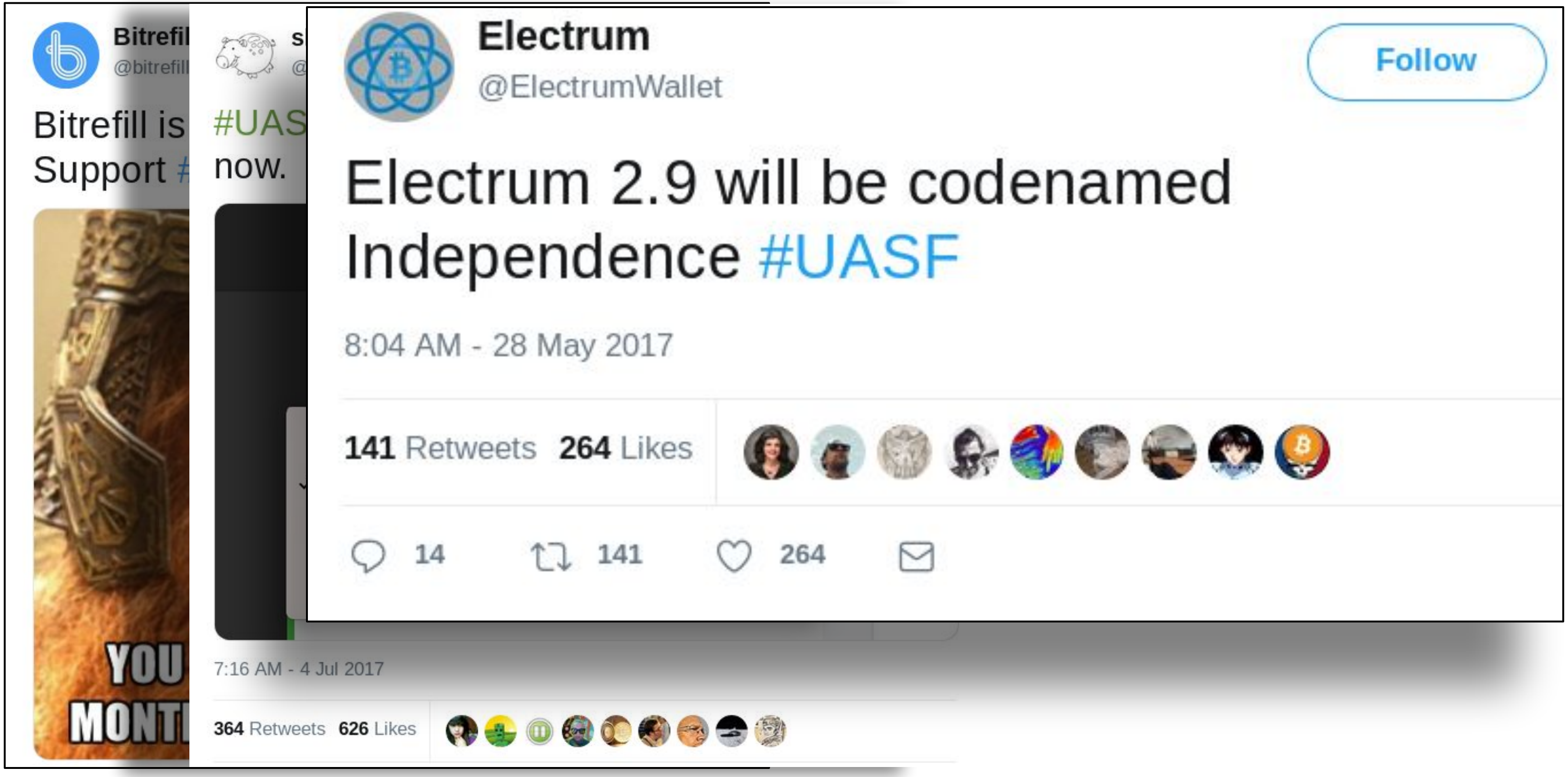
Follow

#UASF on beta-wallet.trezor.io. It's up to you now.

7:16 AM - 4 Jul 2017

364 Retweets 626 Likes

UASF - User Activated Soft Fork



Bitrefil
@bitrefill

Bitrefill is
Support #

s
@

#UAS
now.

7:16 AM - 4 Jul 2017

364 Retweets 626 Likes

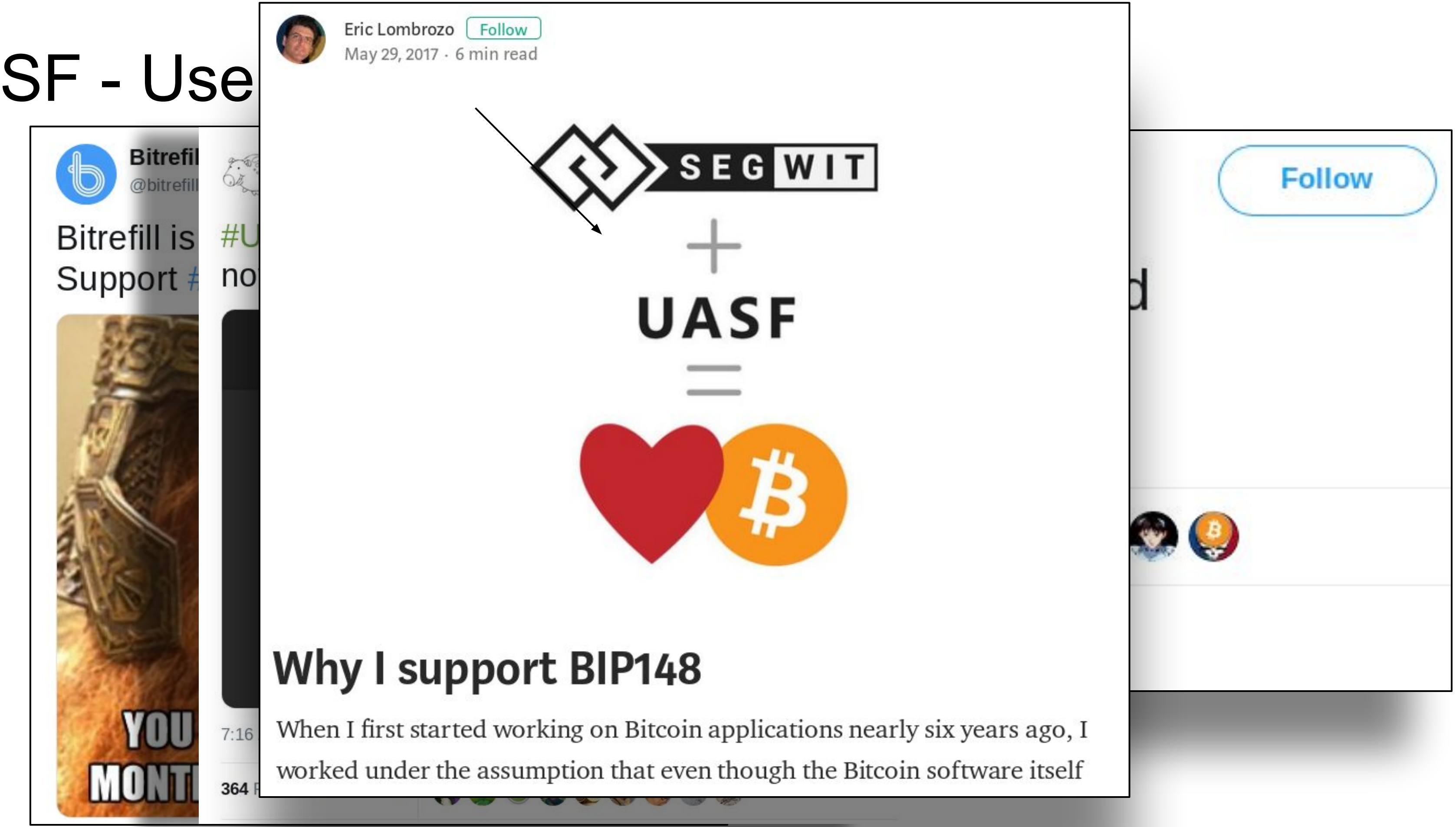
Electrum
@ElectrumWallet

Electrum 2.9 will be codenamed Independence #UASF

8:04 AM - 28 May 2017

141 Retweets 264 Likes

UASF - Use



Bitrefil
@bitrefill

Bitrefill is
Support #

#U
no

7:16

364

Eric Lombrozo Follow
May 29, 2017 · 6 min read

Why I support BIP148

When I first started working on Bitcoin applications nearly six years ago, I worked under the assumption that even though the Bitcoin software itself

Follow

d

UASF - Use

Bitrefil
@bitrefill

Bitrefill is
Support #

#U
no

7:16

364 F

W

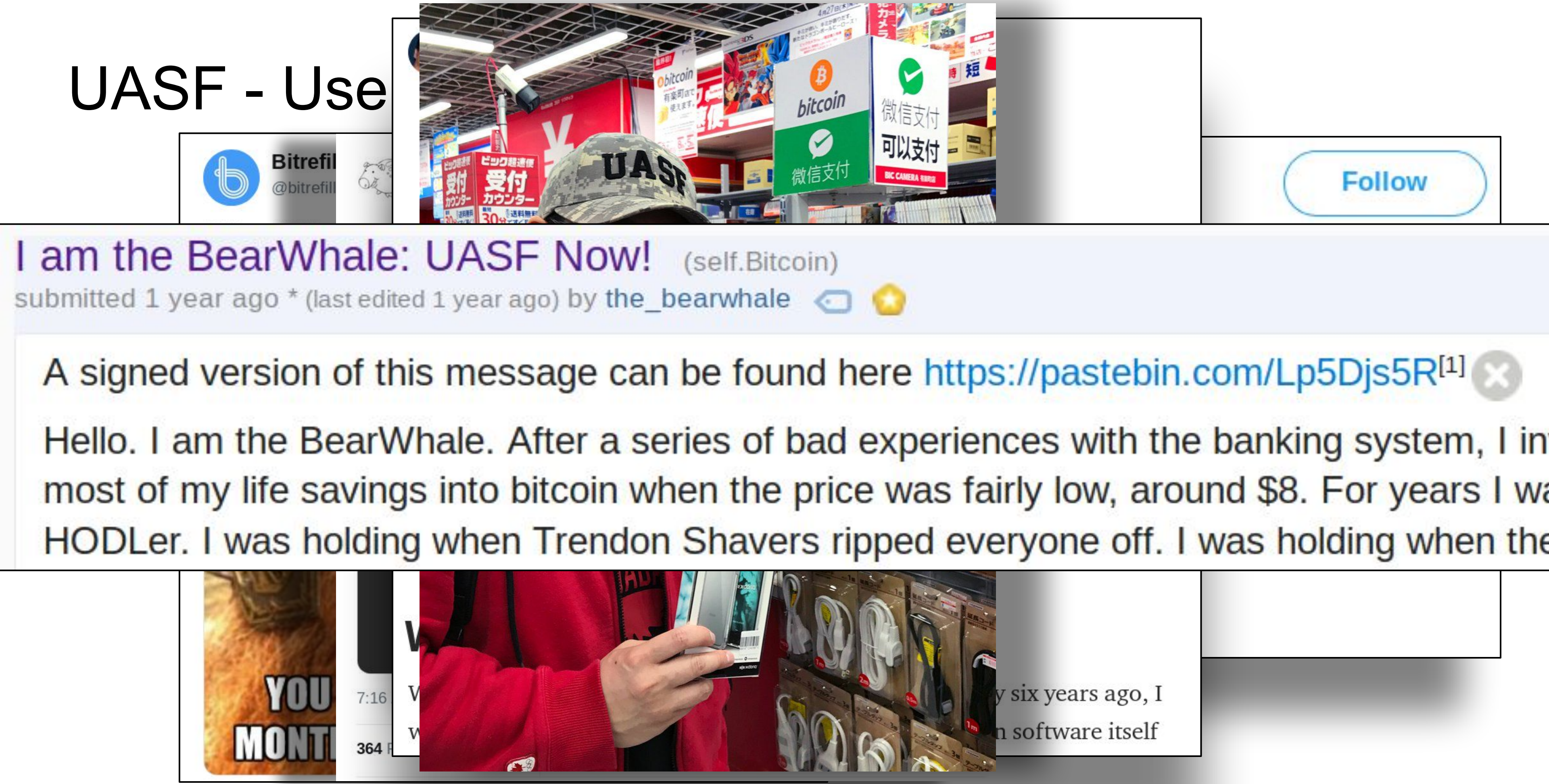
W
w

y six years ago, I
n software itself

Follow

d

UASF - Use



I am the BearWhale: UASF Now! (self.Bitcoin)

submitted 1 year ago * (last edited 1 year ago) by the_bearwhale

A signed version of this message can be found here https://pastebin.com/Lp5Djs5R[1]

Hello. I am the BearWhale. After a series of bad experiences with the banking system, I in most of my life savings into bitcoin when the price was fairly low, around $8. For years I wa HODLer. I was holding when Trendon Shavers ripped everyone off. I was holding when the

UAHF - User Activated Hard Fork

Same concept, but for hardforks.

Examples:

- Bitcoin Cash, born on the same day of segwit lock-in
- Segwit2x (user averted hard fork)

So.. what the hell is Bitcoin?!

So.. what the hell is Bitcoin?!

A social construct, not accurately defined by code or mining power.

Consensus Failure - Chain splits

Different nodes can follow different chains if there is a consensus bug (triggered accidentally or as an attack)

Examples:

- Bitcoin 0.8 introduced an accidental hardfork, switch from BerekelyDB to LevelDB. Fixed soon after with a softfork.
- Recent inflation bug: could have been triggered by a miner, but was not. Fixed with a softfork.

Future of Softforks in Bitcoin

The next softforks will likely be deployed using new segwit script versions.

Might not rely solely on miner signaling any longer, but we’ll see!

Looking forward to Schnorr Signatures, SIGHASH_NOINPUT, and much more!