

# Forks and Upgrade Mechanisms

Or: what the hell is Bitcoin?!!

# Agenda

- BIPs
- 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)

# BIPS

<https://github.com/bitcoin/bips>

## README.mediawiki

People wishing to submit BIPs, first should propose their idea or document to the [bitcoin-dev@lists.linuxfoundation.org](mailto:bitcoin-dev@lists.linuxfoundation.org) mailing list. After discussion, please open a PR. After copy-editing and acceptance, it will be published here.

We are fairly liberal with approving BIPs, and try not to be too involved in decision making on behalf of the community. The exception is in very rare cases of dispute resolution when a decision is contentious and cannot be agreed upon. In those cases, the conservative option will always be preferred.

Having a BIP here does not make it a formally accepted standard until its status becomes Final or Active.

Those proposing changes should consider that ultimately consent may rest with the consensus of the Bitcoin users (see also: [economic majority](#)).

# BIPS

Number	Layer	Title	Owner	Type	Status
1		BIP Purpose and Guidelines	Amir Taaki	Process	Replaced
2		BIP process, revised	Luke Dashjr	Process	Active
8		Version bits with lock-in by height	Shaolin Fry	Informational	Draft
9		Version bits with timeout and delay	Pieter Wuille, Peter Todd, Greg Maxwell, Rusty Russell	Informational	Final
10	Applications	Multi-Sig Transaction Distribution	Alan Reiner	Informational	Withdrawn
11	Applications	M-of-N Standard Transactions	Gavin Andresen	Standard	Final
12	Consensus (soft fork)	OP_EVAL	Gavin Andresen	Standard	Withdrawn
13	Applications	Address Format for pay-to-script-hash	Gavin Andresen	Standard	Final
14	Peer	Protocol Version and User Agent	Amir Taaki, Patrick	Standard	Final

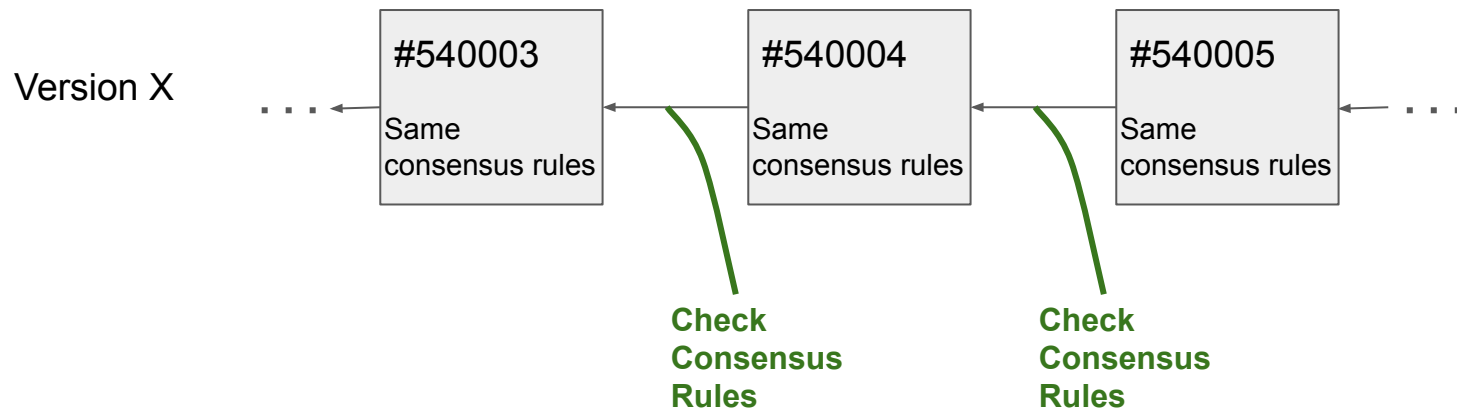
# Consensus Rules

- 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

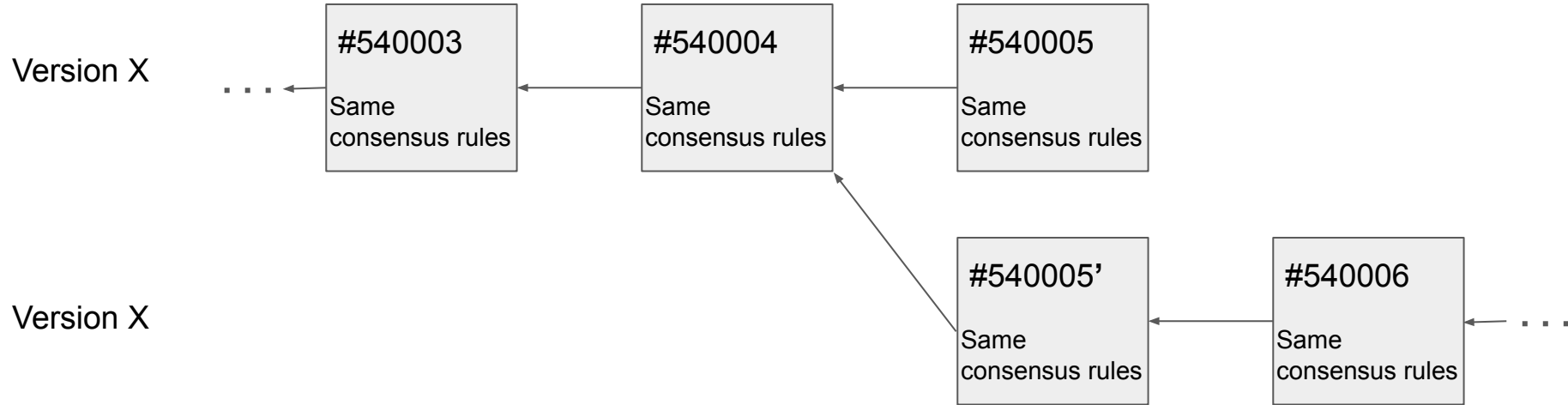
```
281  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script,  
282  {  
284      ...  
299      if (script.size() > MAX_SCRIPT_SIZE)  
300          return set_error(serror, SCRIPT_ERR_SCRIPT_SIZE);
```

# Normal Operation

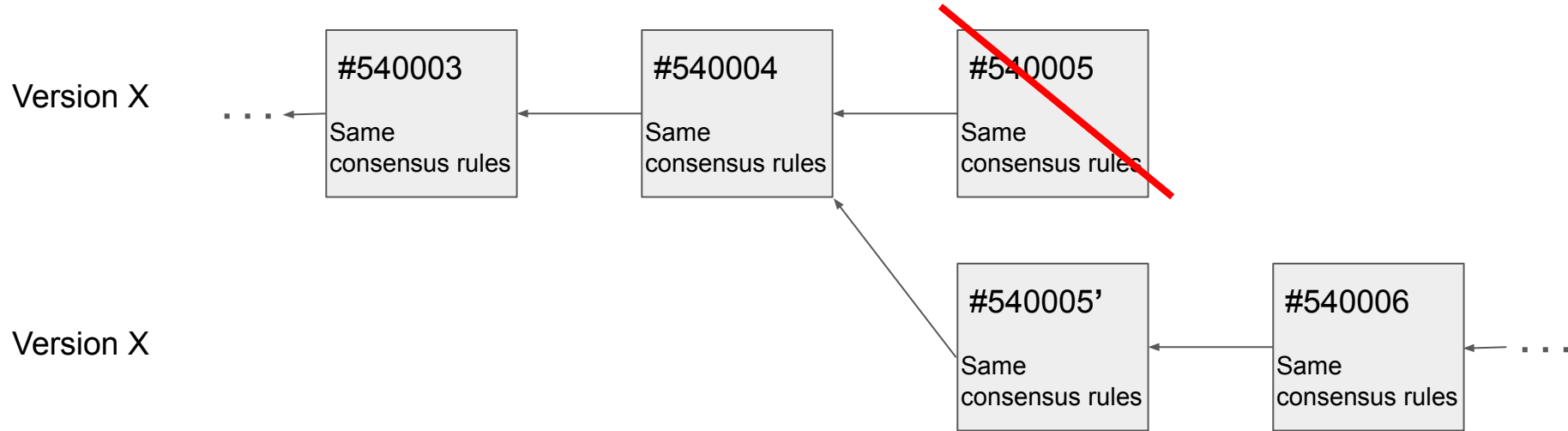




# A fork, but not an upgrade: orphaned blocks



# 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!

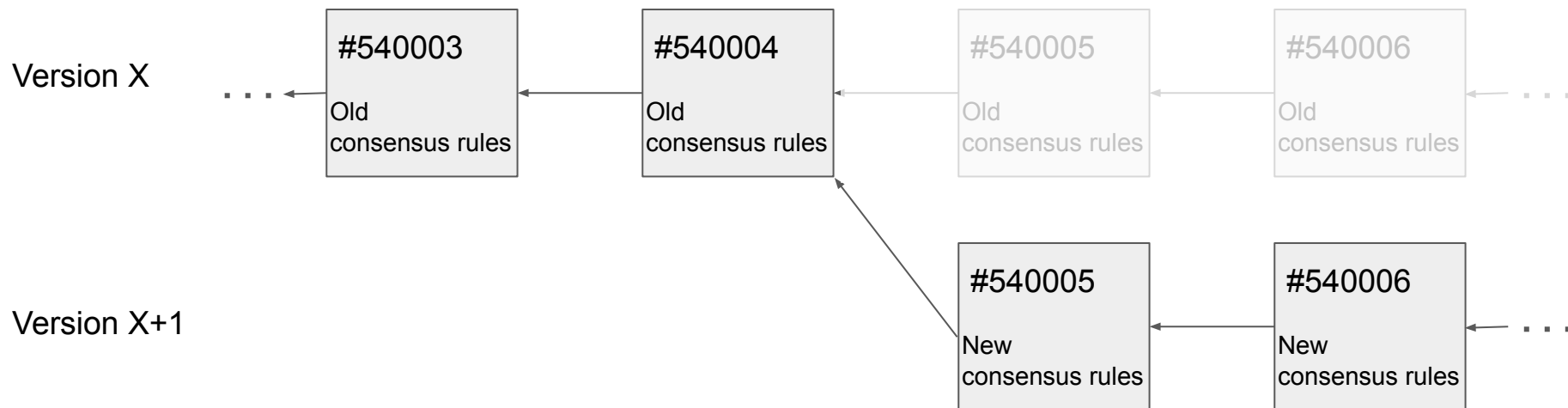
## Examples:

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

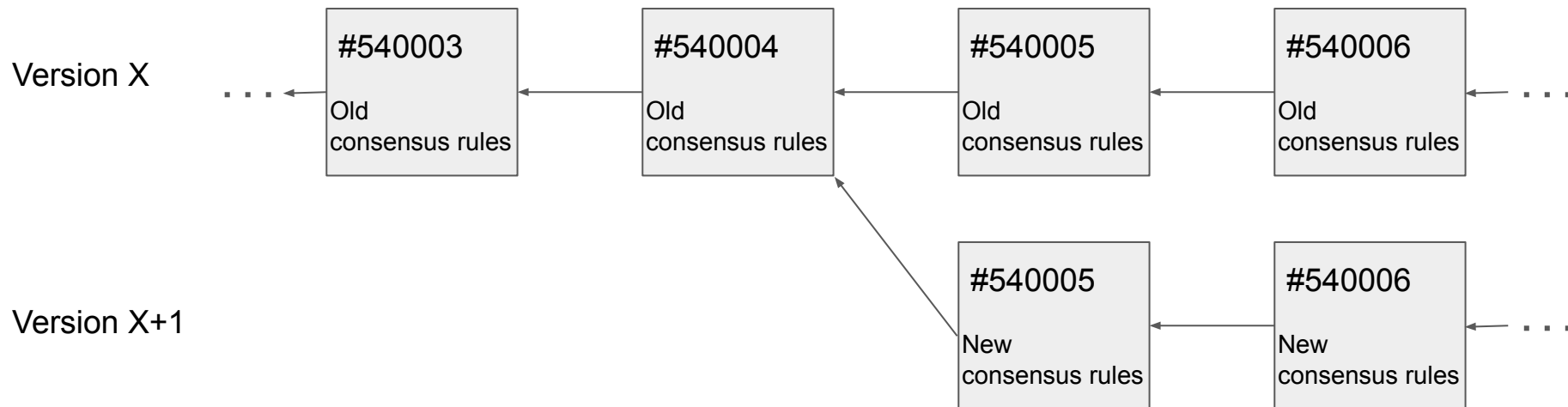
# Hardforks

```
281 bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script,  
282 {  
284     ...  
299     if (script.size() > MAX_SCRIPT_SIZE + 10)  
300         return set_error(serror, SCRIPT_ERR_SCRIPT_SIZE);
```

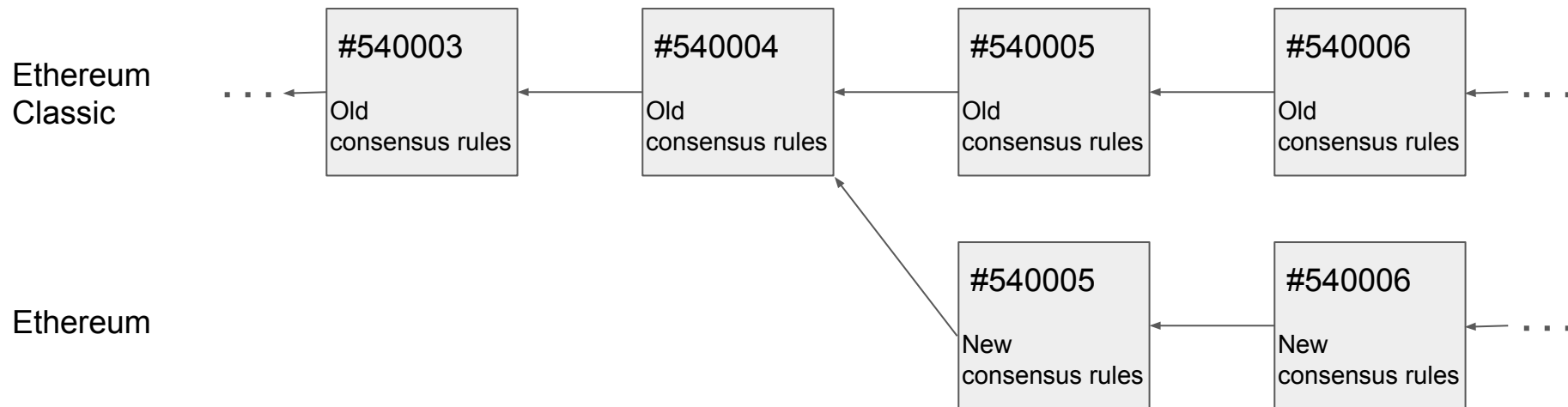
# Hardforks



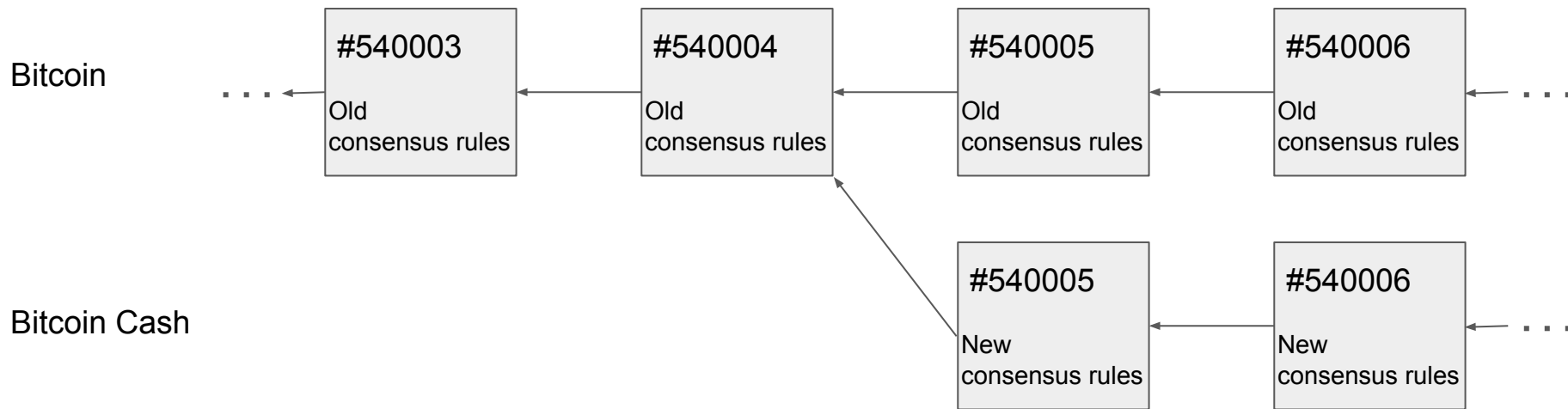
# Hardforks



# Hardforks



# Hardforks





# 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 > 663000) { // expected ~ end of 2020
    // apply new consensus rules, ideally containing replay protection
} else {
    // apply old 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

# Hardfork

- Un  
otl



**Brian Armstrong** ✓

@brian\_armstrong

Follow



.@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...](http://petertodd.org/2016/hardforks...)

10:16 AM - 6 Aug 2016

on the

# 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.  
**They usually don't :(**

# Softforks

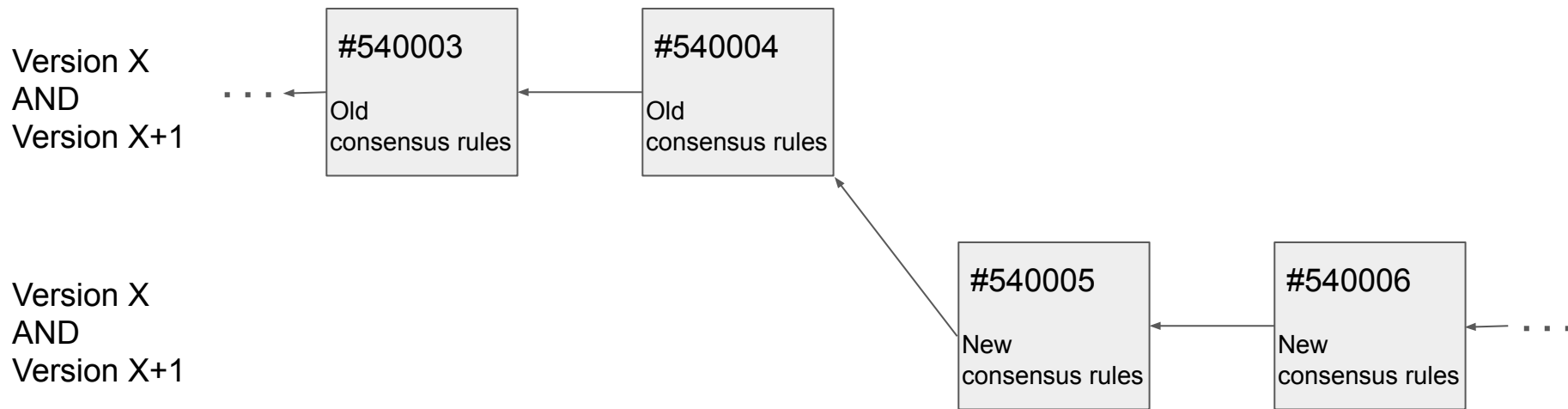
- Softforks add or tighten consensus rules
- Blocks following new rules are **accepted** by old nodes
  - forwards compatible
- **Majority of mining power needs to upgrade. Normal full nodes do not.**

Examples:

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



# Softforks



# 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>`

Arbitrary new scripting language


A thin grey arrow points from the text 'Arbitrary new scripting language' to the '<script>' portion of the witness format.

**Input Script:** *empty*

## Output Script

`<version>`

Provides an easy way to add future script changes through softforks

A thin grey arrow points from the text 'Provides an easy way to add future script changes through softforks' to the '<version>' field of the output script.

`<scriptHash>`



# [bitcoin-dev] Simplicity: An alternative to Script

Russell O'Connor [roconnor at blockstream.io](mailto:roconnor@blockstream.io)

Mon Oct 30 15:22:20 UTC 2017

## Witness

<signature><script>

Arbitrary new scripting language

Input Script: *empty*

## Output Script

<version>

Provides an easy way to add future script changes through softforks

<scriptHash>

# 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

- Loosens rules

vs.

# Softfork

- Tightens rules

# Hardfork

vs.

# Softfork

- Loosens rules
- Risk of chain-split - you have to choose if you want a specific upgrade or not

- Tightens rules
- Single chain, can seamlessly handle either preference

# Hardfork

vs.

# Softfork

- Loosens rules
- Risk of chain-split - you have to choose if you want a specific upgrade or not
- Only one upgrade at a time.

- Tightens rules
- Single chain, can seamlessly handle either preference
- Multiple upgrades at a time.

# Hardfork

vs.

# Softfork

- 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.

- Tightens rules
- Single chain, can seamlessly handle either preference
- Multiple upgrades at a time.
- Previous upgrades not mandatory.

# Hardfork

vs.

# Softfork

- 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.
- Very hard to measure support beforehand.  
Hard to roll out.

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

# Hardfork

vs.

# Softfork

- 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.
- Very hard to measure support beforehand.  
Hard to roll out.
- Easy to measure support after.

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



# Hardfork

vs.

# Softfork

- 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.
- Very hard to measure support beforehand.  
Hard to roll out.
- Easy to measure support after.

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

## 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.

From the BitMex write-up (<https://blog.bitmex.com/bitcoins-consensus-forks/>):

Miners [did not upgrade fast enough](#), so the evaluation point was delayed until 15 March. Users running 0.6.0 rc1 who did not upgrade for the delay activated the softfork early and got [stuck on block 170,060](#) when an invalid transaction, according to their nodes, was mined. After activation, problems were caused as the remaining 45% of miners produced invalid blocks for several months after the softfork

# 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.
  - 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.
  - 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

ASICBOOST

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



# UASF - User Activated Soft Fork

**Bitrefill**  
@bitrefill

**slush**  
@slushcz

Follow

Bitrefill is supporting #UASF on [beta-wallet.trezor.io](https://beta-wallet.trezor.io). It's up to you now.



**TREZOR**

Bitcoin UASF (BIP148) wallet

- Testnet wallet
- Bitcoin wallet
- ✓ Bitcoin UASF (BIP148) wallet
- Dash wallet
- Litecoin wallet
- Zcash wallet
- Ethereum wallet


YOU MONT


7:16 AM - 4 Jul 2017

364 Retweets 626 Likes

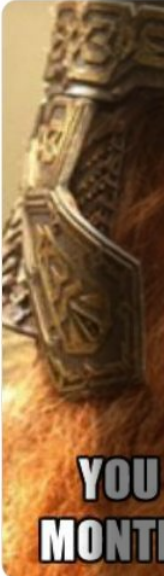


# UASF - User Activated Soft Fork

**Bitrefill**  
@bitrefill


 s @


Bitrefill is  
Support #

**YOU  
MONT**

7:16 AM - 4 Jul 2017

364 Retweets 626 Likes




**Electrum**  
@ElectrumWallet




[Follow](#)

Electrum 2.9 will be codenamed  
Independence **#UASF**

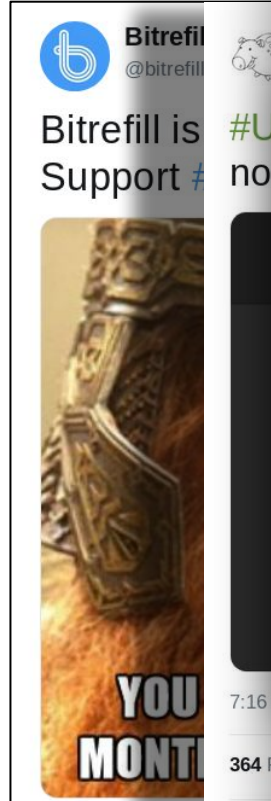
8:04 AM - 28 May 2017

141 Retweets 264 Likes



 14  141  264 

# UASF - Use

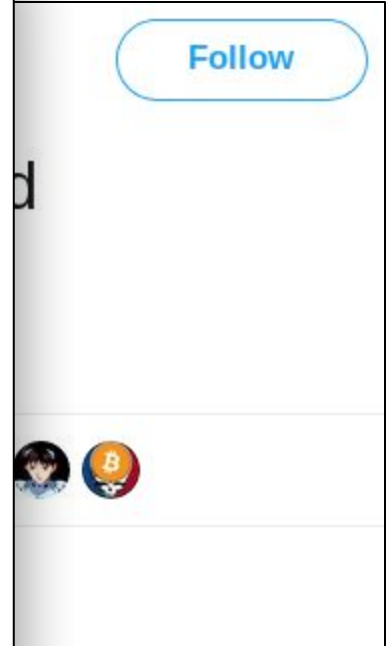


Eric Lombrozo [Follow](#)  
May 29, 2017 · 6 min read

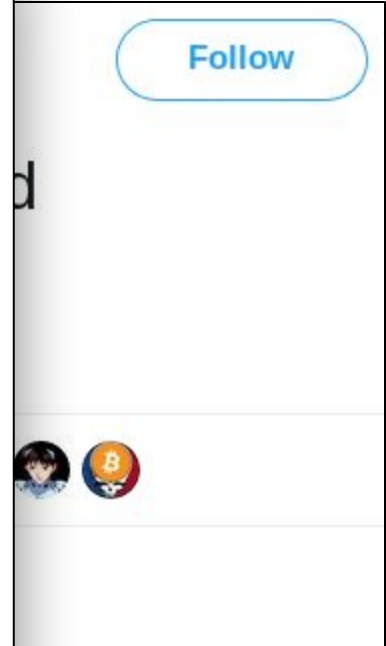
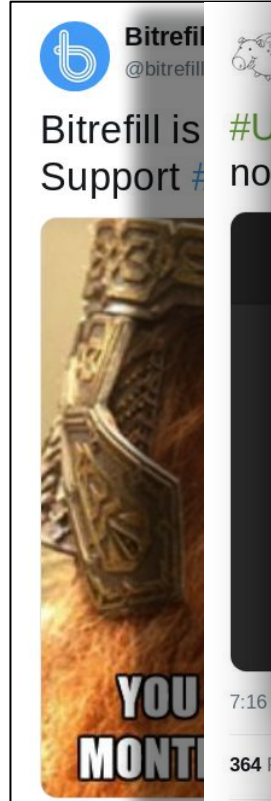
A diagram illustrating the combination of SegWit and UASF. At the top, the SegWit logo (a stylized 'G' inside a square) is shown next to the text "SEG WIT". Below this is a plus sign "+", followed by the text "UASF". Underneath "UASF" is an equals sign "=", which leads to a red heart shape. Inside the heart is a yellow Bitcoin symbol (a circle with a 'B' and two vertical lines).

## 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



# UASF - Use



...y six years ago, I  
...n software itself



# UASF - Use



Bitrefill  
@bitrefill



Follow

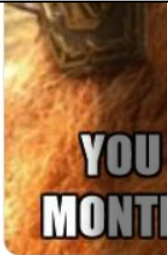
I am the BearWhale: UASF Now! (self.Bitcoin)

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><sup>[1]</sup>

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



7:16

364



...y six years ago, I  
...n software itself

# UASF - User Activated Soft Fork

Segwit locked in August 2017.

Debate over whether miners started signaling because of UASF, or because of the Segwit2x agreement.

Reasonable people can disagree!

# 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 BerkeleyDB 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!

Fin