Replay Protection

- Investor TL&DR[1]
- Replay problems
- Replay protection schemes
  - Ethereum Classic: before and after
  - Bitcoin Cash
  - SegWit2x

---

[1] Things will get quite technical. Feel free to run away after first non-technical slides :-)

Airdrops vs. Forks

- Airdrop
  - "free" coins based on BTC balance[10] at date X
  - safe to ignore, risky to use
- Contentious Hard Fork
  - disagreement on what Bitcoin is
  - not safe to ignore, unless you HODL

---

[10] Assumes you already posses your private keys, as you should.

Airdrop
Free money!? (BCash, BGold, etc)

- 1 BTC on Aug 1 -> 1 BCH
- same private key controls both
  - distrust "official" wallets; assume malware
  - move BTC to fresh wallet first (just in case)
  - privacy (traces on two blockchains)
  - safe to ignore (due to replay protection)

Better safe than sorry. Sooner or later on of these airdrops coins will contain malware. Even without malware, simple incompetence of developers can lead to loss of your bitcoin. Most Bitcoin developers have better things to do than inspect this code. They will write gloating articles explaining what went wrong *after* you lost your Bitcoin.

Wait for well established wallets to support; but they can make mistakes too. Remember Cryptsy.

Contentious Hard fork

SegWit2x may get messy

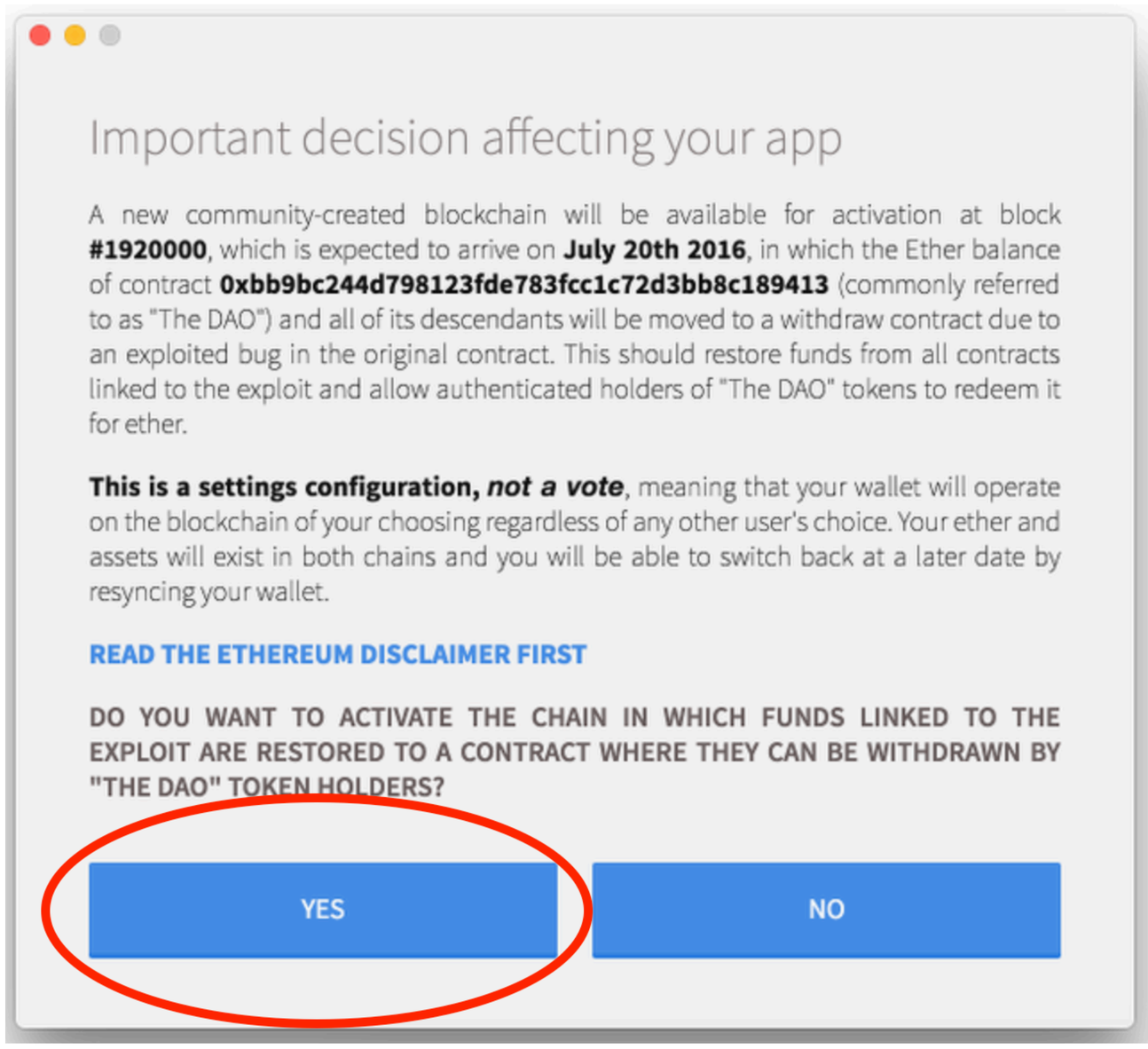
- 1 BTC on Nov ~15 -> 1 BT1 + 1 BT2
- some companies claim BT1 is Bitcoin
- other companies claim BT2 is Bitcoin
- several companies will go back and forth
- no or little replay protection
- never assume companies know what they're doing

Remember The DAO?

- Code is Law!
- $60M ETH stolen from smart contract
- Most developers, holders and miners agreed on need to fork
  - Soft-fork wasn't possible (halting problem)
  - Deadline (not self imposed)

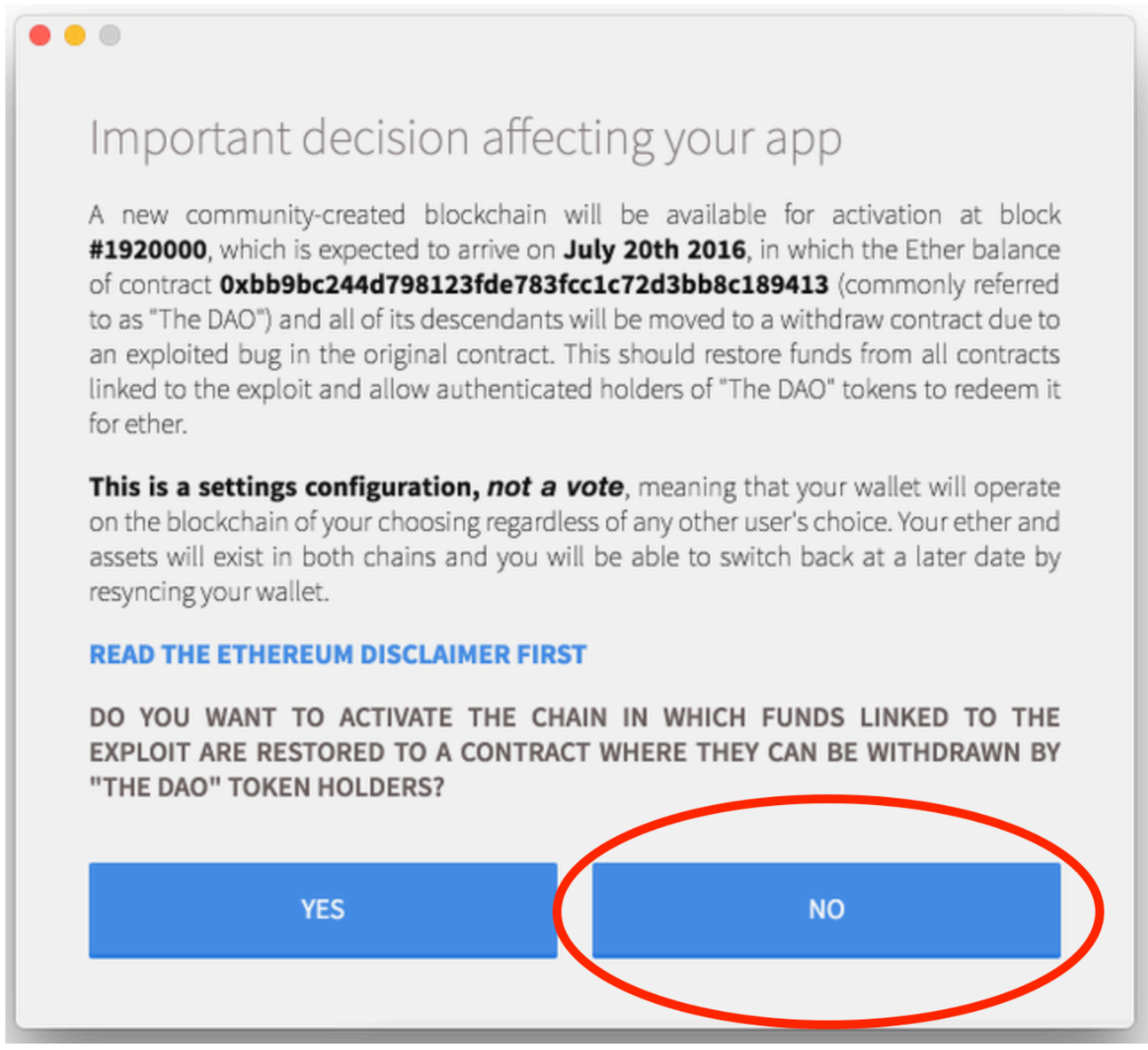
Ethereum Hardfork

- Just click YES



Ethereum Classic

- Just click NO

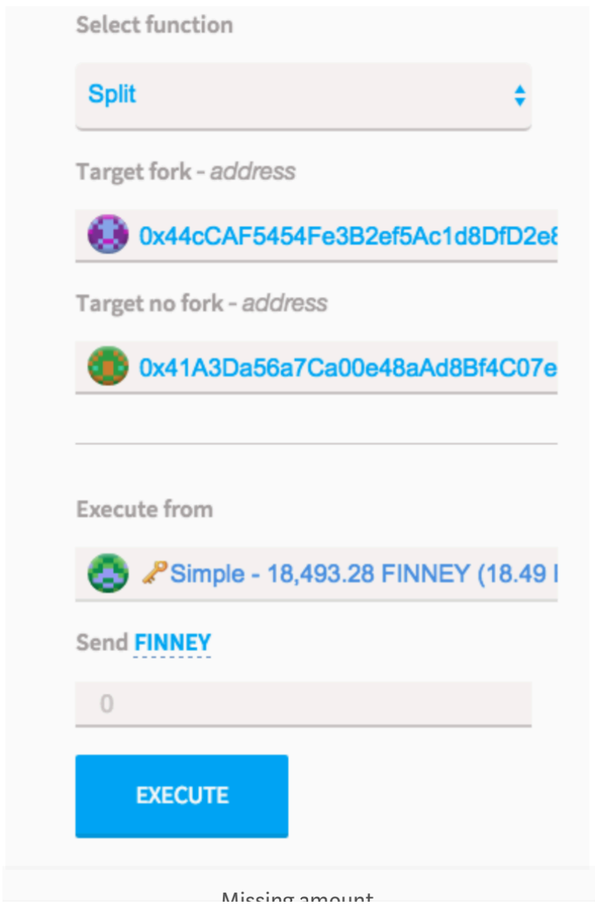


The First Replay Attacks

- Coinbase ETH withdrawals also sent ETC[31]

[31] They had several weeks to prepare. Don't assume companies in this space know what they're doing in all circumstances.

Manual - Split contract



Desktop wallet only follows one chain, but you could label an address as "ETC-only"

Manual - Split contract

```
contract AmIOnTheFork {
    bool public forked = false;
    address constant darkDAO = 0x304a554a310c7e546dfe434669c62820b7d83490;
    function update() {
        if (block.number >= 1920000 && block.number <= 1921200) {
            forked = darkDAO.balance < 3600000 ether;
        }
    }
    function() {
        throw;
    }
}
```


Also: transaction nonce.

Source: https://medium.com/@chevdor/safer-version-of-the-replaysafesplit-smart-contract-a29c347e8a7

Many other versions

Manual - Split contract

Transactions | Contract Source Yes | Read Smart Contract | Comments

⚠ **Warning:** *The compiled contract might be susceptible to ZeroFunctionSelector (very low-severity), DelegateCallReturnValue (low-severity), ECRecoverMalformedInput (medium-severity), SkipEmptyStringLiteral (low-severity), ConstantOptimizerSubtraction (low-severity), IdentityPrecompileReturnIgnored (low-severity), HighOrderByteCleanStorage (high-severity), OptimizerStaleKnowledgeAboutSHA3 (medium-severity), SendFailsForZeroEther (low-severity), DynamicAllocationInfiniteLoop (low-severity), OptimizerClearStateOnCodePathJoin (low-severity) Solidity compiler bugs.*

Contract Source Code Verified

Contract Name:	AmIOnTheFork	Optimization Enabled:	Yes

Manual - 6 easy steps

- Ingredients
  - ETH wallet + ETC wallet
  - 1 teaspoon pure ETH
  - two block explorers
- Procedure
  1. send ETH balance (inc teaspoon)
  2. send ETC balance

EXCHANGE The no-brains trick to separate your ETH and ETC without using a splitter contract or sending all your belongings to Poloniex self.ethtrad

Toegevoegd op 1 jaar geleden * door cryptopascal Dapp Dev

As title mentions, several other ways to split your ETH and ETC exist.

If found this one the most simple, and also the only way to split completely (if you send ETH to a contract, you mostly need to leave some small amount because you do not know the transaction fee beforehand):

Step 1: check whether your really have a balance on both chains:

(replace 0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7 with your address)

ETC: http://gastracker.io/addr/0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7
ETH: http://etherscan.io/address/0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7

Step 2: send some symbolic amount of "pure" ETH or ETC to either one of your addresses

Pure ETH or ETC means ETH or ETC that has been separated already, in other words, sent to you in a transaction that cannot be replayed.

The easiest source is getting it from an Exchange that has split their funds, such as Kraken or Poloniex (setting up an account takes a few minutes at Poloniex, no ID verification required for small amounts). But you can always ask a friend or other people on the forum to send you 0.01 ETH.

In my example I will continue by withdrawing 0.02 ETC from Poloniex, resulting in a 0.01 transaction (for ETC there's a minimum 0.01 transaction fee).

Step 3: check again you really have different balances

ETC: http://gastracker.io/addr/0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7
ETH: http://etherscan.io/address/0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7

So my ETC balance is now higher than my ETH balance because of this transaction:
http://gastracker.io/tx/0xd37682144cf295cadd3f531b1ca89eed8ead31f14e843cbbafccec113598b832

Step 4: Completely EMPTY the account with the HIGHEST balance to a fresh address

In my case I have 0.01 *more* ETC than I have ETH at the address 0xC2aa74847e86EDFdd3F3dB22f8A2152fEee5b7F7. If I transfer that full amount to a fresh address, that transaction **cannot be replayed** for ETH. The easiest way to do that, is to use the "Send Everything " option in the Mist Wallet.

Important: only send to an address you yourself control! Assume for example you sent it to me as a payment, I could quickly top up the lower balance on the other chain so the transaction becomes replayable again and I get your money on the other chain too :-)

Check your address again in the ETH and ETC block explorers to check that one address is now empty, and the other one still has the original amount.

Step 5: Completely EMPTY the other account to a DIFFERENT fresh address

This step is to be on the super safe side. My example again: because my ETC balance is now empty, no ETH transactions from my original address can be replayed for ETC. On the other hand, no transactions from my new ETC address can be replayed, because the corresponding ETH balance is empty.

There are a few edge cases however: imagine that you use your old address as a kind of user account, and you send a transaction to vote for the statement "This chain is more beautiful than the other chain".

Then *theoretically* someone could come in on the other side, send some coins to *your* old empty address (on that other chain), and replay your transaction, so it would look like you voted for the same statement on the other side (DAH HORROR!).

So in conclusion, to be on the safe side: empty also the second account to a fresh address (and of course: different from your fresh new address on the other chain!), using the "Send Everything " option in the Mist Wallet

Step 6: Enjoy your cleanly split lives on both sides

Extra note: how to run clients for both ETH and ETC

1. if you are a Mist user, then the easiest option is probably to install version 0.8.1 twice, *under two separate user accounts on your Mac or Windows machine*. Answer Yes on the startup question on the DAO fork for one user and no for the other.

2. If you have a hosted wallet, you are at a loss. AFAIK there are no wallets yet that allow you to send transactions at both chains. Some people have already forked MyEtherWallet and host a version that connects to an ETC backend, see https://np.reddit.com/r/EthereumClassic/comments/4u4o61/call_for_action_what_can_i_do_to_help_ethereum/d5r8boi and

This can still go wrong if an attacker sends you a teaspoon on the other chain quickly enough

Automatic - EIP 155

Specification

If `block.number >= FORK_BLKNUM` and `v = CHAIN_ID * 2 + 35` or `v = CHAIN_ID * 2 + 36`, then when computing the hash of a transaction for purposes of signing or recovering, instead of hashing only the first six elements (ie. nonce, gasprice, startgas, to, value, data), hash nine elements, with `v` replaced by `CHAIN_ID`, `r = 0` and `s = 0`. The currently existing signature scheme using `v = 27` and `v = 28` remains valid and continues to operate under the same rules as it does now.

- another hard fork (Spurious Dragon, Nov 2016)
- opt-in, but wallets use by default
- same address format

Several months later...

Each hard fork needs to decide if they want to add replay protection, so this requires guessing if it's going to be contentious.

Bitcoin Cash

- initially opt-in
- last minute change to mandatory
- same address format
- SIGHASH_FORKID and BIP43

Bitcoin Cash

BIP143: new signature algorithm

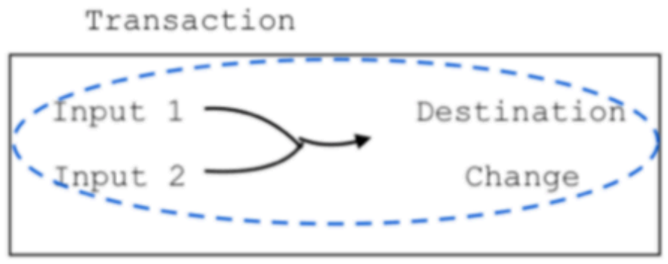
- covers value of the input being spent
- solves quadratic hashing
- must be combined with SegWit
- BCH uses BIP143 without SegWit[40]
  - BCH tx invalid on BTC chain

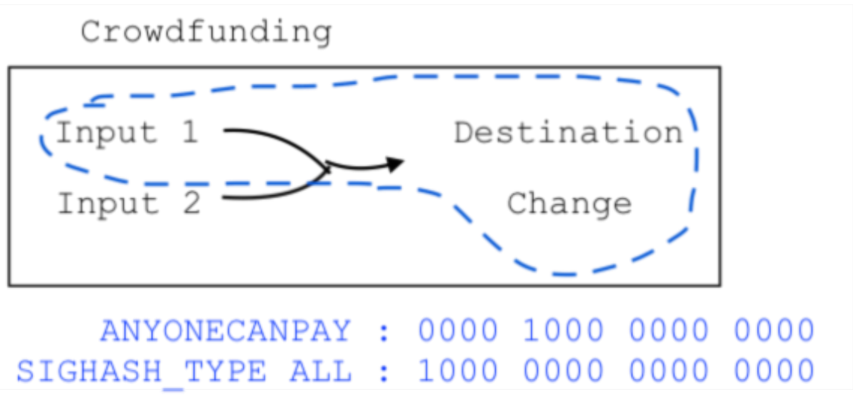
[40] this gives them some of the benefits of SegWit

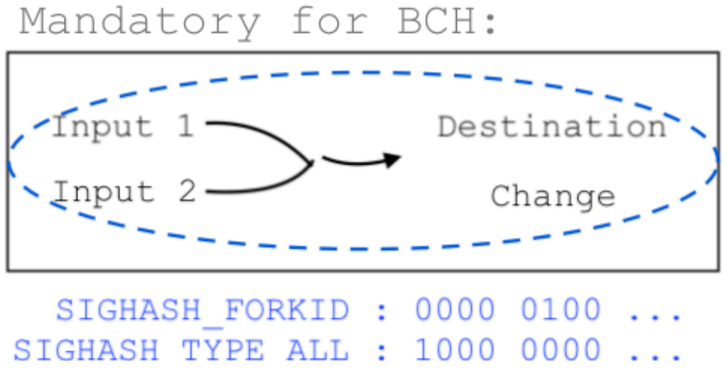
"The BIP143 signature generating algorithm covers the value of the input being spent, which simplifies the design of air-gapped light-weight wallets and hardware wallets." - https://bitcoincore.org/en/segwit*wallet*dev/

Bitcoin Cash
SIGHASH_FORKID

- mandatory for BCH
- valid but non-standard for BTC
- makes BTC transactions invalid on BCH chain
- combined with BIP143: protection both ways
- illustration in next slide







the 0x01 at the end of a normal transaction is SIGHASH_ALL.

https://github.com/bitcoinbook/bitcoinbook/blob/second*edition*print_1/ch06.asciidoc#signature-hash-types-sighash

https://github.com/Bitcoin-ABC/bitcoin-abc/blob/master/src/script/interpreter.h#L26

https://github.com/Bitcoin-UAHF/spec/blob/master/replay-protected-sighash.md#sighash-type (is this more recent?)

(?) SigHash field is 4B when you sign it, but it gets truncated to the last byte when you serialise the signature.

Bitcoin Cash - Ledger device

```
21   #define SIGHASH_ALL 0x01
22  +#ifdef COIN_BITCOIN_CASH
23  +#define SIGHASH_FORKID 0x40
24  +#endif
25
26   unsigned short btchip_apdu_hash_sign() {
27       unsigned long int lockTime;

87
88                if (((N_btchip.bkp.config.options &
89                      BTCHIP_OPTION_FREE_SIGHASHTYPE) == 0)) {
90  +#ifdef COIN_BITCOIN_CASH
91  +                 if (sighashType != (SIGHASH_ALL | SIGHASH_FORKID))
    {
92  +                     sw = BTCHIP_SW_INCORRECT_DATA;
93  +                     goto discardTransaction;
```


Screw specs, just read wallet source code.

The changes needed on the device to support it were pretty simple: https://github.com/LedgerHQ/blue-app-btc/commit/6fcd9d27f078e0a035019562adc4a875d9aa27e6, i.e. don't sign a transaction if SIGHASH_FORKID isn't set.

Chrome plugin magic happens here: https://github.com/LedgerHQ/ledger-wallet-chrome/blob/1.9.6/app/libs/btchip/btchip-js-api/BTChip.js#L1674-L1675

Bitcoin Cash

	Good miner	Naughty miner	Stupid miner
BCH tx			
BTC chain	-	-	invalid: BIP143 w/o SegWit
BCH chain	standard[42]	-	-
BTC tx			
BTC chain	standard	-	-
BCH chain	-	-	invalid: FORKID

[42] IsStandard is a subset of all valid transactions. Non-standard transactions are not relayed or mined by default, but, a "naughty miner" could just mine it, making replay protection ineffective.

Bitcoin Gold

- TBD
- Addresses will start with G (A for SegWit)

https://github.com/BTCGPU/BTCGPU/issues/1717#issuecomment-339801030

SegWit2x Constraints[44]

1. minimal changes to software of participants[45]
2. capture light weight wallets[46]
3. nice to have; mostly a gesture to Core
4. limited development and review resources[45]
5. (?) avoid hard-fork with BU

---

[44] self imposed

[46] light weight clients can just inspect block 494784

[45] most participants are adding non-protocol level replay protection

SegWit2x Constraints

	Good miner	Naughty miner	Stupid miner
Unupgraded wallet			
1x	standard	-	-
2x	standard	-	-
BT1-only legacy & SegWit			
1x	standard	-	-
2x	-	-	invalid
BU	-	-	non-standard
BT2-only legacy & SegWit			
1x	-	-	invalid
2x	standard	-	-
BU	standard	-	-

SegWit2x
1x-only using magic address

- 3Bit1xA4apyzgmFNT2k8Pvnd6zb6TnwcTi
- manual, no wallet change needed
- UTXO "spam"
- phishing "tutorials"
- BU support (using standardness)

Not all wallet support sending to multiple addresses

^ https://github.com/BitcoinUnlimited/BitcoinUnlimited/pull/790

SegWit2x
1x-only using magic address

	Good miner	Naughty miner	Stupid miner
B1X tx (to 3Bi...)			
1x chain	standard	-	-
2x chain	-	-	invalid: address
B2X tx (not to 3Bi...)			
1x chain	replay	standard	-
2x chain	standard	-	-

https://github.com/btc1/bitcoin/commit/a3c41256984bf11d95a560ae89c0fcbadfbe73dc

SegWit2x

1x-only using OP_RETURN

- OP_RETURN RP=!>1x
- no UTXO spam

https://github.com/btc1/bitcoin/pull/134

SegWit2x
1x-only using OP_RETURN

	Good miner	Naughty miner	Stupid miner
BT1 (OPRETURN RP=!>1x)			
1x chain	standard	-	-
2x chain	-	-	invalid: OP_RETURN
BT2 (no OP_RETURN)			
1x chain	replay	standard	-
2x chain	standard	-	-

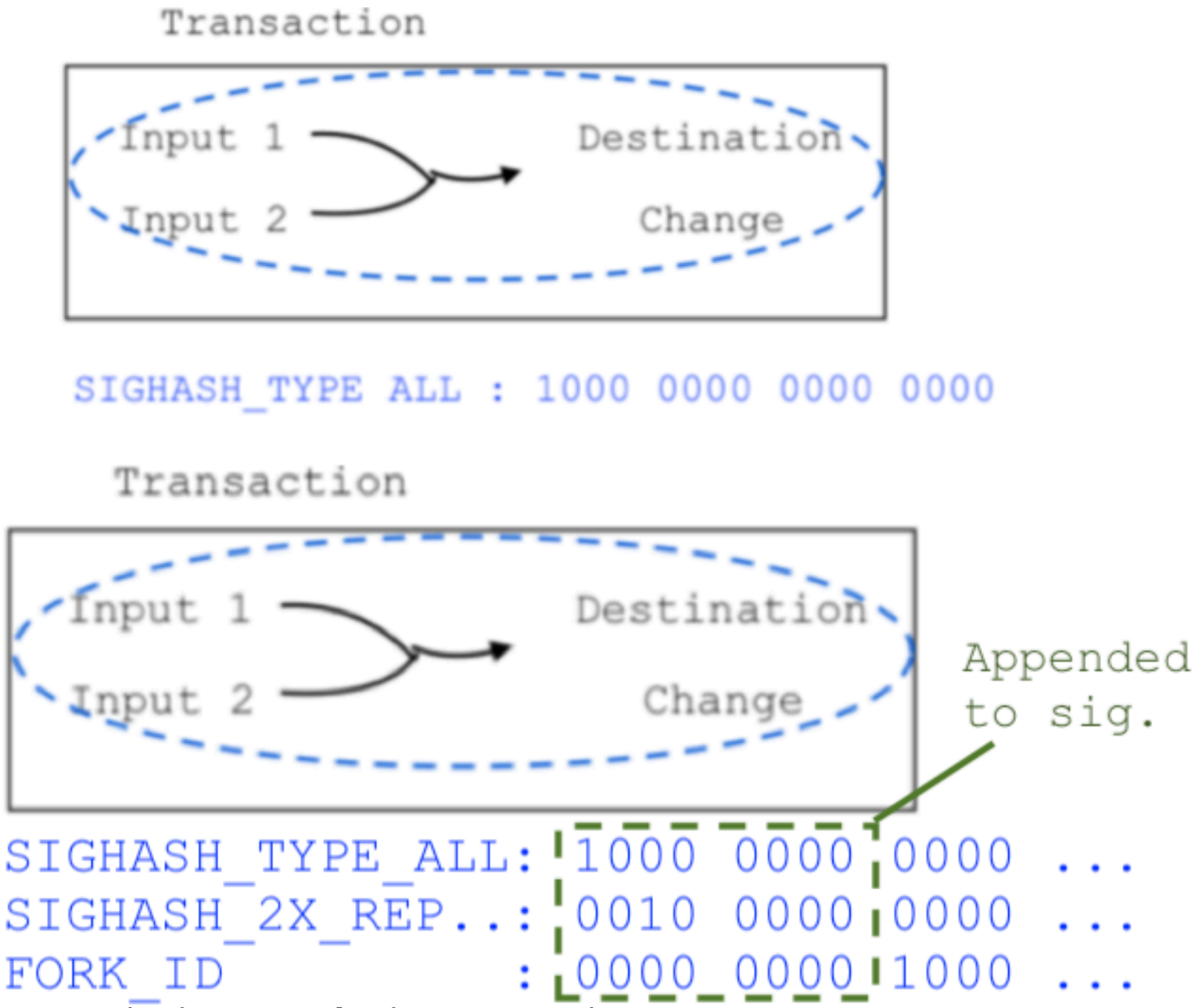
SegWit2x
2x-only - SIGHASH magic

- New: **SIGHASH_2X_REPLAY_PROTECT** (4)
- Sets bit 8 in pre-image (BCH used bit 6)
- Bit 8 isn't appended to signature
  - Core node consider signature invalid
- hard-fork relative to BU

```
186 -bool static IsDefinedHashtypeSignature(const valtype &vchSig) {
187       if (vchSig.size() == 0) {
188           return false;
189       }
190       unsigned char nHashType = vchSig[vchSig.size() - 1] & (~
    (SIGHASH_ANYONECANPAY));
191 -     if (nHashType < SIGHASH_ALL || nHashType > SIGHASH_SINGLE)
192           return false;
193
194       return true;
```

```
186 +bool static IsDefinedHashtypeSignature(const valtype &vchSig,
    const bool allowReplay) {
187       if (vchSig.size() == 0) {
188           return false;
189       }
190       unsigned char nHashType = vchSig[vchSig.size() - 1] & (~
    (SIGHASH_ANYONECANPAY));
191 +     if (nHashType < SIGHASH_ALL
192 +         || (allowReplay && nHashType > SIGHASH_2X_REPLAY_PROTECT)
193 +         || (!allowReplay && nHashType > SIGHASH_SINGLE))
194           return false;
195
196       return true;
```

```
1272 + +     if (fReplayProtection && (nHashType & 0x1f) ==
     SIGHASH_2X_REPLAY_PROTECT)
1273 +         nHashType |= (1U << 8);
```



https://github.com/btc1/bitcoin/pull/131

Set bit 8 if SIGHASH2*X*REPLAY_PROTECT is set and we're past the fork block

https://bitcoin.stackexchange.com/questions/48108/why-are-sighash-flags-signed-as-4-bytes-when-only-1-byte-is-included-in-the-tran

https://bitcoin.stackexchange.com/questions/61706/where-is-sighash-type-serialized-in-segwit

SegWit2x

2x-only: SIGHASH magic

	Good miner	Naughty miner	Stupid miner
BT1 tx (no magic)			
1x chain	standard	-	
2x chain	replay	standard	-
BT2 tx (SIGHASH magic)			
1x chain	-	-	invalid signature
2x chain	standard	-	-

This is quite different from the BCH approach.

Spoonnet Replay protection

Somewhat similar to the above.

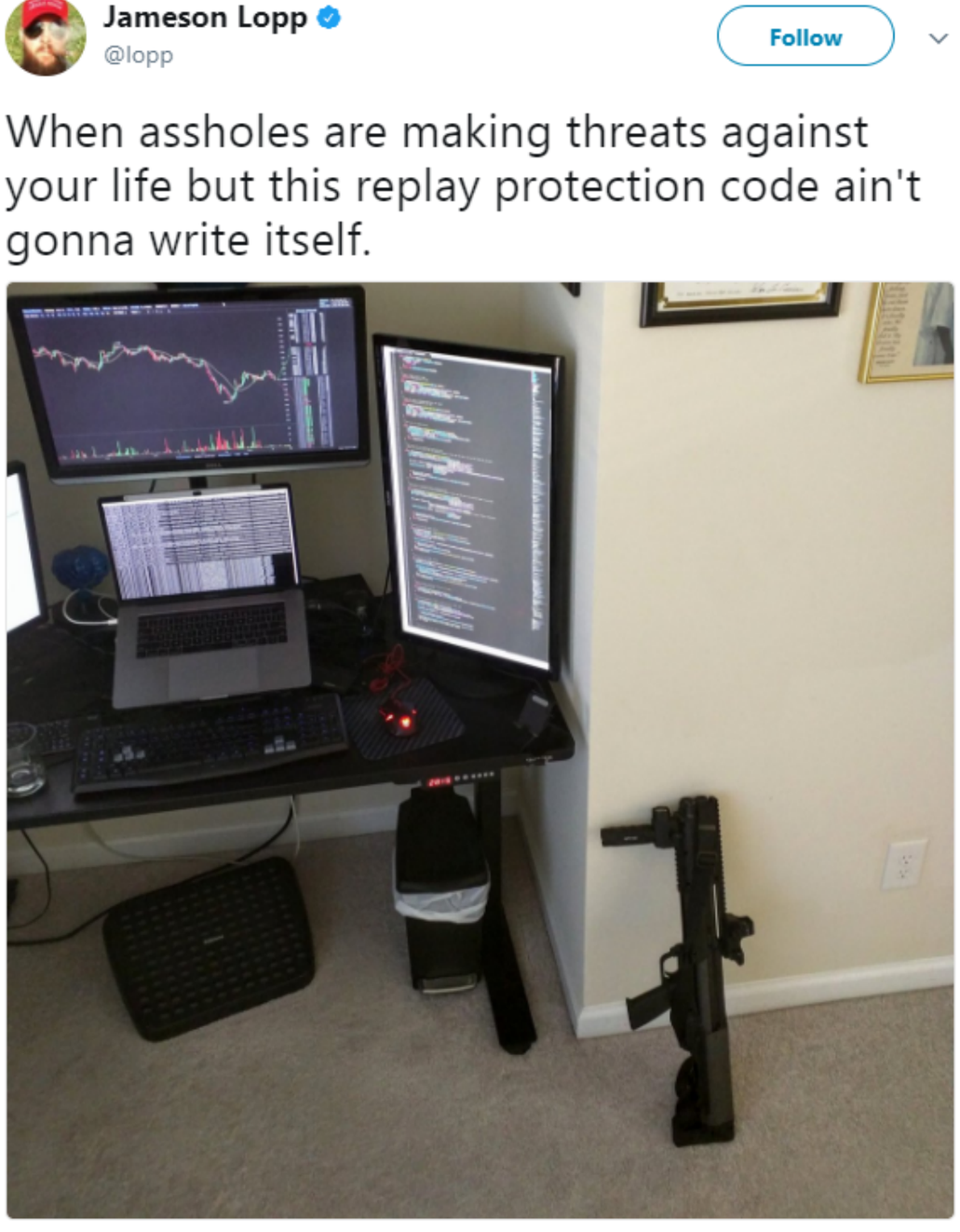
- uses nVersion [50]
- hardfork network version bit is 0x02000000
- 0x02000000 is added to the nHashType
- leaves serialized **SIGHASH_TYPE** alone

---

[50] "A tx is invalid if the highest nVersion byte is not zero, and the network version bit is not set" (bitcoin-dev list)

SegWit2x - Unprotected?

- HODL!
- UTXO mixing
- nLockTime
- >1 MB transaction
- Or just use a custodial service :-(
  - Exchanges
  - Split services



Easiest thing to do during fork is to not use Bitcoin for a while, but not everyone has that luxery.

Custodial wallets and exchanges can take care of the splitting. They can split customer funds in batches, saving money.

Maybe a 1 MB transaction can be done via a trustless mixer? But how to guarantee it's bigger than 1 MB?

Some other options here: https://github.com/btc1/bitcoin/issues/34#issuecomment-339573341

UTXO Fairy Dust

- Ask miner: coinbase tx unique for each side[70]
- Service can split using other method
- (paid) API with anyone-can-spend UTXO's?
- Wallet coin selection must include these inputs[71]

---

[70] natural, organic replay protection, but can't be done until 100 blocks after the fork

[71] proof-of-replay-protection?

nLockTime - 4 easy steps

nLockTime: not mined or relayed[80] before block N

1. generate two addresses (A1, A2)
2. check which chain moves faster (e.g. H2 > H1)
3. sign tx[81] to A2 with H1 < nLockTime < H2
4. send to A1 w/o nLockTime[82]

---

[80] IsStandard() rules

[81] Bonus: use RBF (unpredictable fee market)

[82] wait until confirmed, try again if needed

https://en.bitcoin.it/wiki/Timelock

H1: block height of 1x chain,
H2: block height of 2x chain

nLockTime - problems

- wallet must monitor both chains
- need to wait for gap in block height
- sweep is bad for privacy
- must wait for step 4, risks:
  - reorg (e.g. intentional wipeouts)
  - fees in BTC terms > balance
- receiving new unsplit funds

Only works while one side of fork has a big enough lead. Can't be used immediately after fork

This is hard to do manually, but also hard to automate for non-custodial wallets. User needs to come back several times, lots of edge cases to handle in UI.

When receiving new funds, wallet must reason if those funds are already replay protected, or its coin selection must always include coins that are known to be protected.

We'll learn all sorts of new problems as people start losing their money.

> 1MB

- valid on 2x, invalid on 1x
- non-standard (coordinate with miners)
- expensive (easier for a service)
- maybe using CoinJoin?

Other Fork problems

- Address reuse
- Privacy
- Unconfirmed transactions
- Price feeds
- Bad decisions by judges
- ...

Replay protection is not the only problem...

Thanks

Dump your 1x / 2x coins here:

- Slides: slideshare.net/provoost
- Blog: medium.com/provoost-on-crypto

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512