Theory and Practice of Improving Bitcoin

CS 251(p): Bitcoin and Crypto Currencies
Part 15, Lecture 15: Sidechains
Nov 14, 2016

Greg Maxwell and Pieter Wuille

The Theory and Practice of Improving Bitcoin

Topics

- Challenges in advancing Bitcoin
- An abstraction of Bitcoin and its historical evolution
  - Balances vs UTXO
  - Addresses / Bitcoin Script as a predicate / P2SH / MAST
  - Computation vs Verification and Bitcoin as a court
- Stupendous applications of this abstraction
  - Sidechains and Cut-through
- Ongoing and future efforts

On authority in Bitcoin

We work on Bitcoin but there is no “official Bitcoin”, no “CEO of Bitcoin”, no guaranteed “roadmap” and this is inherent to Bitcoin's value.

No so different from the Linux Kernel or internet standards:

> "*Instead of a roadmap, there are technical guidelines. Instead of a central resource allocation, there are persons an companies who all have a stake in the further development of the Linux kernel, quite independently from one another: People like Linus Torvalds and I don’t plan the kernel evolution. We don’t sit there and think up the roadmap for the next two years, then assign resources to the various new features. That's because we don’t have any resources. The resources are all owned by the various corporations who use and contribute to Linux, as well as by the various independent contributors out there. It's those people who own the resources who decide.*"
>
> *- Andrew Morton on the kernel process*

On authority in Bitcoin *(cont)*

- So... there is no “right” or “wrong” way to understand Bitcoin.
- But there are more useful or less useful ways to understand it.
- We'll be presenting a mental model for Bitcoin and improvements to it which has been collaborative developed over six years and is largely shared by the majority of the technical community but not widely known elsewhere.
- We'll explain it in the historical order that it was developed in, and then talk about its applications and implications.
- We believe this model is highly useful.

What is Bitcoin?

- Bitcoin is P2P electronic cash that is valuable over legacy systems because of the monetary autonomy it brings to its users through decentralization. To be attractive as a money Bitcoin must have a reasonable level of fairness, accessibility, security, fungibility, and privacy.
  - This is challenging because there is competition and trade-offs between these important objectives.
- E.g. Bitcoin is secured by users receiving the entire state of the system and verifying all changes to it. This is at odds with privacy.
- Through clever technology we seek to maximize these qualities without compromising the rest.

Challenges in Advancing Bitcoin

- Traditional money systems require trust at all levels, this trust is costly to maintain and is unpredictably violated
- Bitcoin's solution
  - Replace most of the trust with a system of mechanically enforceable rules
- Nature protests: a copy of data is as good as the original, information doesn't have “owners”
  - Money needs controlled supply and ownership
  - People are good at ignoring rules (*when they want*)

Challenges in Advancing Bitcoin

Good news:
Bitcoin employs cryptography and economics to deliver a system where rules have true force, even against popular will

Bad news:
This created a system where a fixed set of protocols / algorithms were in charge

- If mankind had perfect engineering, perfect foresight, and universal values, it might be okay
- But we don't: mistakes are made, needs change, and people sometimes earnestly have contradictory demands.

Challenges in Advancing Bitcoin

- Some have sought to create new functionality by starting brand new cryptocurrencies
- The value of a money-like good comes from acceptance – it's practically all network effect
- A speculative race around "creating money": bad incentives and no natural stopping point: **foocoin→barcoin→bazcoin→barfcoin**
- The reboot is left with the same problem
- We wish people luck, but don't think this is a sustainable way to build new technology

Balances vs UTXO

- Bitcoin's consensus does not have accounts, it tracks transaction outputs. “Coins”
- Accounts are toxic to privacy and once you use as many accounts for privacy they provide little value but still incentivize harming privacy.
- Replay is naturally resolved by the UTXO model.
  - Alice pays Bob and Charlie but before the transactions are confirmed Alice wants to update her payment to Bob, without accidentally preventing payment to the Charlie
- “Functional model” the network has a state and a transaction decides if a new state is permitted to follow. Results in a compact running state (1.6GB today vs 85GB, ”pruning”).

Bitcoin 'Addresses' (P2PKH)

- Secp256k1 pubkeys (without compression) are 512-bits long:
  - hard to communicate but only ~128-security
- Pragmatic solution: give the sender a 160-bit hash of the pubkey, and reveal the full pubkey when spending.
  - 160-bit against second-premages > 128-bit DLP security.
- Thankfully, Bitcoin is programmatic== transactions pay to a 'script'
  - “DUP HASH256 <20-bytes> EQUALVERIFY CHECKSIG”

What is script actually doing?

> “The nature of Bitcoin is such that once version 0.1 was released, the core design was set in stone for the rest of its lifetime. Because of that, I wanted to design it to support every possible transaction type I could think of. [...] The solution was script, which generalizes the problem so transacting parties can describe their transaction as a predicate that the node network evaluates. The nodes only need to understand the transaction to the extent of evaluating whether the sender's conditions are met. ***The script is actually a predicate. It's just an equation that evaluates to true or false***. Predicate is a long and unfamiliar word so I called it script.” (Bitcoin's creator, June 2010)

- Script is not a language for writing computation: It's a language for writing spendability conditions.
- Why restrict addresses' indirection to simple 1-pubkey scripts?

P2SH

- Give the sender the hash of the entire script, and reveal the script at spending time. (BIP16, activated April 2012)
- 20 bytes (+ checksum) address covers any possible script
- Small privacy improvement and major interface improvement
  - It usually isn't any of your business how I manage *my* coins.
- A practical necessity for making multisignature available.
- About 10% of all Bitcoin are secured using P2SH today.

MAST or What if we go deeper?

We can use this principle that P2SH used recursively:

Merkelized Abstract Syntax Tree

	Traditional	**P2SH**	**MAST**
Contract	(A or B,C,D)	H("A or B,C,D")	H(H(A) or H(B,C,D))
Evidence	sigA	(A or B,C,D), sigA	(A, H(B,C,D)), sigA
or	sigBCD	(A or B,C,D), sigBCD	(H(A), (B,C,D)), sigBCD

- *(the reason we're calling these contract and evidence will become clear soon)*

- Realization: we don't need to reveal the full spending conditions.
- Significant efficiency and privacy gain

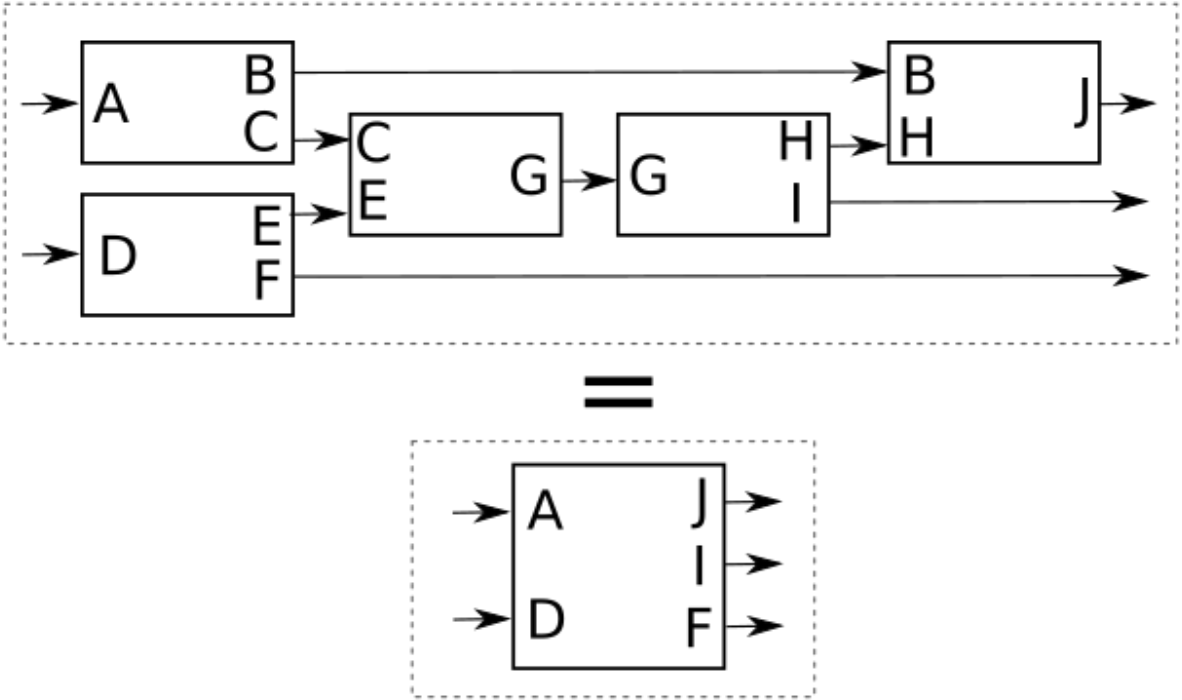
Validation vs Computation

- There is a more fundamental insight here: Proving that conditions are satisfied does not require everyone to re-evaluate them.
  - Question: would we ever want a square root operation in script?
  - Answer: Probably not, any script that needs (A == sqrt(B)) can be rewritten as (A^2 == B)
- Same thing for mul/dev, matrix mult vs matrix inv, ... P vs NP!
- More generically: Why have the whole world recompute what you already know? You only need to provide evidence for correctness.

Bitcoin as a Court

- With MAST we can write any complicated contract as (everybody agrees) OR (contract), and not usually reveal the full contract.
  - Why would all parties usually agree? All know the other parties can choose to reveal the full conditions, and prove their case
- Analogy: We don't have a policeman following everyone at all times to make sure nothing goes wrong: Not efficient or private.
  - We have courts with parties trusted to make judgments
- But we don't go conduct all our business before a Court
  - We arrange our business so that we can present it before a court if needed, in the most efficient way possible.
  - The existence of the court creates honesty even when it isn't used

Transaction Cut-through



Cooperative transaction merging reducing seven signatures to two

(or five to one with aggregation)

Transaction Cut-through (cont.)

- Truly eliminates transactions from having to be preserved in the chain without depending on third parties.
- Back to the Court analogy: cooperating parties do not need to go to court at all.
- Can be done in Bitcoin since day one but requires coordination.
  - Cut-through + Confidential transactions + Aggregation = Mimblewimble
  - Cut-through + Payment Channels + Hash locked transactions = Lightning
- Can we move non-cooperating parties to a 'lower court' or a more flexible court?

Evolution of Side chains

- Late 2010, proposals for using the Bitcoin network for generalized non-Bitcoin applications: naming, property registries, etc. which were potentially complementary but had high resource costs
- This was discouraged because the cost of these uses would be an externality on Bitcoin users, and that Bitcoin users might get increasingly tyrannical about limiting the size of the chain to preserve the ability participate
- Separate networks with separate fates could share resources (e.g. merged mining) and support trustless asset exchange (via atomic swap transactions). But cannot use this for actual Bitcoins!

Evolution of Side chains: One-way peg

- Oct 2013, A. Back describes the challenge of deploying new technology without rebooting the network effect in “is there a way to do bitcoin-staging?” and proposes a *one-way* peg:
- “Betacoin” nodes, a separate network (sidechain), watch the Bitcoin network and award coins when Bitcoins are provably destroyed
- Results in a flag day free, choice preserving upgrade path
- Appears viable but only if people are absolutely sure the upgrade is the one true path forward: no way to go back, betacoins inherently less valuable.

Evolution of Side chains: CoinWitness

- Aug 2013 we observe that efficient proof for general computation allows users to trustlessly assign Bitcoins to the control of external systems for improved scalablity / flexibility / privacy.
- The external systems can be of any construction so long as their operation is transcript verifiable
- Public key verifies a program that verifies the transcript and releases the Bitcoins if the program accepts
- Works for probabilistic-consistent systems too, using timeouts
- Requires verifying the system under a proof, not realistic right now

SPV two way pegged sidechains

- Bitcoin supports verifying a payment with small amounts of communication: Simplified Payment Verification *(bitcoin.pdf §8)*
  - There is a security tradeoff with SPV: It trusts miners to verify the history, stronger non-partitioning assumption
- With SPV the rules are cheap enough to plausibly verify with Bitcoin script.
- The result is the “two-way peg” described in the sidechains whitepaper

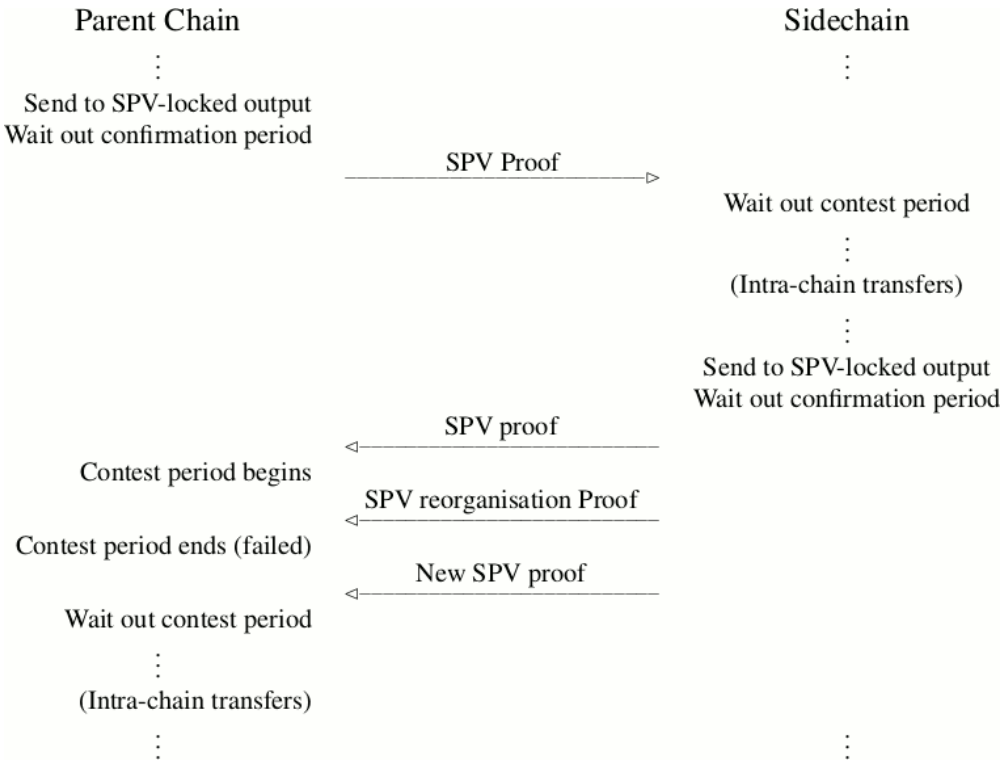
Enabling Blockchain Innovations with Pegged Sidechains

Adam Back, Matt Corallo, Luke Dashjr,
Mark Friedenbach, Gregory Maxwell,
Andrew Miller, Andrew Poelstra,
Jorge Timón, and Pieter Wuille∗†

2014-10-22 (commit 5620e43)

SPV 2WP transaction phases

Whitepaper describes the 2WP in terms of a symmetric mechanism where both chains use the same procedure to verify the other



2WP: Drawbacks and risks

- Complexity *(implementing all this correctly takes work)*
- Reorg beyond timeout is unrecoverable
  - Long reorgs also “effectively unrecoverable” in Bitcoin, but...
  - Without subsidy and with only SPV security on the peg the security model looks more like Bitcoin in 20 years than now
- Risk of “soft-fork”- breaking the isolation to boost security
- Potential for increased load on miners reducing decentralization if merged mined

Succinct Zero Knowledge

- With MAST, justice may come at the price of privacy
- There are systems that allow you to cryptographically convince someone of the truthfulness of any predicate without showing the details.
- They can have sublinear or constant proof size. (regardless of how big the inputs or are how long normal way of verifying)
- Logical conclusion to the thinking we've presented:
  - The ideal system would have a block that exposes a UTXO delta and a proof. The content and number of transactions would be completely private.

Succinct Zero Knowledge *(cont.)*

- Current succinct ZKP are barely practical:
  - Prover operates about as fast as a 10Hz CPU.
  - Verifier still magnitudes slower than signatures verify
  - The most practical systems (SNARKS) have new and serious security assumptions, and require trusted setup!
- Doesn't sound like something the Bitcoin community would be eager to adopt directly.
- We we expect this to improve. They also often inspire other ideas (sidechains) and there are ways to use them without Bitcoin supporting them directly.

ZKP without the leap of faith

- A zero knowledge contingent payment (ZKCP) is a way that one person can purchase specific information from another in a secure way, proposed in 2011:
  - Bob wants to buy data X and has a program P that returns true only on valid X. Alice knows X but only wants to give it to Bob if he pays. No one wants to go add support for P to Bitcoin (it's slow or app specific.)
- Alice gives Bob values Y,Z and proves using a ZKP that Y==H(K) && Enc(X,K)==Z && P(X) == True; all without using the blockchain.
- Bob pays, requiring Alice sign and reveal some K such that H(K)==Y OR a timeout passes and Bob signs.
- Demonstrated by Greg Maxwell and Sean Bowe in Jan 2016.

Summarizing the model evolution

- Least sophisticated understanding: The blockchain as a global computer (everybody executes fixed programs)
  - Predicate Script: Programmable contracts
  - Addresses and P2SH: Only publish at the time of execution
  - MAST: Do not publish the full contract
  - Cut-through: Do not publish anything if all parties agree
  - Sidechains *(and Lightning)*: Arbitrate details elsewhere
  - ZKP (and SNARKS): Only ever publish proofs
- The blockchain as a verifier (only store evidence necessary to prove validity)

Sidechains in practice: Elements Alpha

A Bitcoin testnet sidechain

Released June 10th, 2015

Deterministic Peg

Asset Issuance

Relative Locktime [Now Bitcoin!]

Segregated Witness [Bitcoin soon?]

Script Enhancements

Amount under Sig [Bitcoin soon?]

Federated Consensus

Confidential Transactions

https://github.com/ElementsProject/elements

Why Segregated Witness?

- A Bitcoin can be spent if some input is provided which makes its assigned script returned true: Script is a predicate
- All solutions should be equally good
- What happens if you can take a true script solution and modify it and get another one? “Malleability”!
  - Third parties can change transaction IDs
  - Potentially breaks multi-step contracts, or even just spending an unconfirmed transaction

Why Segregated Witness?

- A witness is a specific value that constitutes a concrete proof for an existential claim
- Bitcoin doesn't care *why* the scriptPubkey accepted, just that it does
- If you're not verifying the history, instead trusting it blindly, you don't care about the witnesses, but they are 2/3 of the data.
  - But you have to fetch it to verify transaction hashes

Segregated Witness

- Originally deployed in Elements Alpha sidechain.
  - Highly disruptive design which would require invasive changes to every wallet and Bitcoin application.
- Discovered a way to make it completely backwards compatible
  - Leave existing signatures empty, add an extra 'witness area' outside the transaction that new nodes use.
  - Use P2SH to hide the new script from sending wallets

Segregated Witness

- Specified in BIP141, BIP143, BIP144, and BIP145
- Created an opportunity to make a number of related improvements:
  - Signature covers value
  - Script Versioning
  - Quadratic signature hashing cost
  - Externality reducing cost function for UTXO impact
- The last of these results in an effective capacity increase

Segregated Witness

- Spec was done in March 2016, active on testnet since May (and four earlier specialized testnets since January)
- In Bitcoin Core 0.13.1 (October 27th), quorum sensing on the network starting at the next retarget (on the 18th).
- Will activate when 95% hashpower signals they will enforce during a 2016 block window.

Ongoing and Future efforts

Mining incentives and block propagation

- Backlogs and pay-forwards
  - Preserving the incentive to move forward as fee goes away
- Pre-consensus
  - Agree in advance what will get mined to remove transfer and validation from the block race which reduces a centralization pressure
- Mempool reconciliation (min bandwidth for max agreement)
- FIBRE (forward error correction to globally sync blocks with consistently the lowest possible latency)
- Adaptive, incentive compatible resource controls

Ongoing and Future efforts

- MimbleWimble
  - Strong privacy and improved scalability with boring cryptographic tools
- Powerful crypto external to Bitcoin (ZKCP, private solvency proofs)
- Lightning and other payment channel techniques
- ZKP verification of whole blockchain rule sets
- Formal validation on Bitcoin consensus logic and underlying software tools

Ongoing and Future efforts

Future script systems

- MAST (several prototypes done so far)
- Signature aggregation (~30% reduction in effective txn sizes)
  - Schnorr aggregation that requires interaction
  - BLS aggregation that doesn't
- More expressive predicate languages which are highly agreeable with formal verification

Thanks for your time.

Greg Maxwell
DE47 BC9E 6D2D A6B0 2DC6 10B1 AC85 9362 B041 3BFA

Pieter Wuille
133E AC17 9436 F14A 5CF1 B794 860F EB80 4E66 9320