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)

Abstract

Since the introduction of Bitcoin[Nak09] in 2009, and the multiple computer science and electronic cash innovations it brought, there has been great interest in the potential of decentralised cryptocurrencies. At the same time, implementation changes to the consensus-critical parts of Bitcoin must necessarily be handled very conservatively. As a result, Bitcoin has greater difficulty than other Internet protocols in adapting to new demands and accommodating new innovation.

We propose a new technology, *pegged sidechains*, which enables bitcoins and other ledger assets to be transferred between multiple blockchains. This gives users access to new and innovative cryptocurrency systems using the assets they already own. By reusing Bitcoin's currency, these systems can more easily interoperate with each other and with Bitcoin, avoiding the liquidity shortages and market fluctuations associated with new currencies. Since sidechains are separate systems, technical and economic innovation is not hindered. Despite bidirectional transferability between Bitcoin and pegged sidechains, they are isolated: in the case of a cryptographic break (or malicious design) in a sidechain, the damage is entirely confined to the sidechain itself.

This paper lays out pegged sidechains, their implementation requirements, and the work needed to fully benefit from the future of interconnected blockchains.

Contents

1 Introduction

David Chaum introduced digital cash as a research topic in 1983, in a setting with a central server that is trusted to prevent *double-spending*[Cha83]. To mitigate the privacy risk to individuals from this central trusted party, and to enforce fungibility, Chaum introduced the *blind signature*, which he used to provide a cryptographic means to prevent linking of the central server's signatures (which represent coins), while still allowing the central server to perform double-spend prevention. The requirement for a central server became the Achilles' heel of digital cash[Gri99]. While it is possible to distribute this single point of failure by replacing the central server's signature with a *threshold signature* of several signers, it is important for auditability that the signers be distinct and identifiable. This still leaves the system vulnerable to failure, since each signer can fail, or be made to fail, one by one.

In January of 2009, Satoshi Nakamoto released the first widely used implementation of peer-to-peer trustless electronic cash[Nak09], replacing the central server's signature with a consensus mechanism based on proof of work[Bac02], with economic incentives to act cooperatively. Bitcoin tracks payments by aggregating them into *blocks*, each with an associated *blockheader*, which cryptographically commits[1] to: the contents of the block, a timestamp, and the previous blockheader. The commitments to previous headers form a *blockchain*, or *chain*, which provides a well-defined ordering for transactions.

We observe that Bitcoin's blockheaders can be regarded as an example of a *dynamic-membership multi-party signature* (or *DMMS*), which we consider to be of independent interest as a new type of group signature. Bitcoin provides the first embodiment of such a signature, although this has not appeared in the literature until now. A DMMS is a digital signature formed by a set of signers which has no fixed size. Bitcoin's blockheaders are DMMSes because their proof-of-work has the property that anyone can contribute with no enrolment process. Further, contribution is weighted by computational power rather than one threshold signature contribution per party, which allows anonymous membership without risk of a *Sybil attack* (when one party joins many times and has disproportionate input into the signature). For this reason, the DMMS has also been described as a solution to the Byzantine Generals Problem[AJK05].

Because the blocks are chained together, Bitcoin's DMMS is cumulative: any chain (or chain fragment) of blockheaders is also a DMMS on its first block, with computational strength equal to the sum of the strengths of the DMMSes it is composed of. Nakamoto's key innovation is the aforementioned use of a DMMS as a signature of *computational power* rather than a signature of *knowledge*. Because signers prove computational work, rather than proving secret knowledge as is typical for digital signatures, we refer to them as *miners*. To achieve stable consensus on the blockchain history, economic incentives are provided where miners are rewarded with fees and subsidies in the form of coins that are valuable only if the miners form a shared valid history, incentivising them to behave honestly. Because the strength of Bitcoin's cumulative DMMS is directly proportional to the total computational power contributed by all miners[Poe14a], it becomes

---

[1]A *commitment* is a cryptographic object which is computed from some secret data, but does not reveal it, such that the data cannot be changed after the fact. An example of a commitment is a *hash*: given data x, one can publish $H(x)$ where H is a hash function, and only later reveal x. Verifiers can then confirm that the revealed value is the same as the original value by computing $H(x)$ themselves.

infeasible for a computational minority to change the chain. If they try to revise the DMMS-secured ledger, they will fall behind and be continually unable to catch up to the moving target of the progressing consensus blockchain.

Because the miners do not form an identifiable set, they cannot have discretion over the rules determining transaction validity. Therefore, Bitcoin's rules must be determined at the start of its history, and new valid transaction forms cannot be added except with the agreement of every network participant. Even with such an agreement, changes are difficult to deploy because they require all participants to implement and execute the new rules in exactly the same way, including edge cases and unexpected interactions with other features.

For this reason, Bitcoin's objective is relatively simple: it is a blockchain supporting the transfer of a single native digital asset, which is not redeemable for anything else. This allowed many simplifications in the implementation, but real-world demands are now challenging those simplifications. In particular, current innovation is focused around the following areas:

1. There are trade-offs between scalability and decentralisation. For example, a larger block size would allow the network to support a higher transaction rate, at the cost of placing more work on validators — a centralisation risk.

   Similarly, there are trade-offs between security and cost. Bitcoin stores every transaction in its history with the same level of irreversibility. This is expensive to maintain and may not be appropriate for low value or low-risk transactions (*e.g.* where all parties already have shared legal infrastructure in place to handle fraud).

   These trade-offs should be made for each transaction, as transactions vary widely in value and risk profile. However, Bitcoin by construction supports only a "one size fits all" solution.

2. There are many more trade-offs for blockchain features. For example, Bitcoin's script could be more powerful to enable succinct and useful contracts, or could be made less powerful to assist in auditability.

3. There are assets besides currencies that may be traded on blockchains, such as IOUs and other contracts, as well as smart property [Sza97].

4. There is a risk of monoculture: Bitcoin is composed of many cryptographic components, any one of whose failures could cause a total loss of value. If possible, it would be prudent not to secure every bitcoin with the same set of algorithms.

5. New technology might enable new features not imagined when Bitcoin was first developed. For example, privacy and censorship-resistance could be improved by use of cryptographic accumulators[Mou13], ring signatures[vS13], or Chaumian blinding[Cha83].

6. Even if there is a pressing need to do so, there is no safe upgrade path for Bitcoin, in the sense that all participants must act in concert for any change to be effected. There is consensus amongst Bitcoin developers that changes to Bitcoin must be done slowly, cautiously, and only with clear assent from the community.

4

The fact that functionality must be broadly acceptable to gain adoption limits participants' personal freedom and autonomy over their own coins. Small groups are unable to implement features, such as special-purpose script extensions[jl213], because they lack broad consensus.

An early solution to these problems with Bitcoin has been the development of alternate blockchains, or *altchains*, which share the Bitcoin codebase except for modifications to address the above concerns. However, implementing technical changes through the creation of independent but essentially similar systems is problematic.

One problem is infrastructure fragmentation: because each altchain uses its own technology stack, effort is frequently duplicated and lost. Because of this, and because implementers of altchains may fail to clear the very high barrier of security-specific domain knowledge in Bitcoin[Poe14c], security problems are often duplicated across altchains while their fixes are not. Substantial resources must be spent finding or building the expertise to review novel distributed cryptosystems, but when they are not, security weaknesses are often invisible until they are exploited. As a result, we have seen a volatile, unnavigable environment develop, where the most visible projects may be the least technically sound. As an analogy, imagine an Internet where every website used its own TCP implementation, advertising its customised checksum and packet-splicing algorithms to end users. This would not be a viable environment, and neither is the current environment of altchains.

A second problem is that such altchains, like Bitcoin, typically have their own native cryptocurrency, or *altcoin*, with a floating price. To access the altchain, users must use a market to obtain this currency, exposing them to the high risk and volatility associated with new currencies. Further, the requirement to independently solve the problems of initial distribution and valuation, while at the same time contending with adverse network effects and a crowded market, discourages technical innovation while at the same time encouraging market games. This is dangerous not only to those directly participating in these systems, but also to the cryptocurrency industry as a whole. If the field is seen as too risky by the public, adoption may be hampered, or cryptocurrencies might be deserted entirely (voluntarily or legislatively).

It appears that we desire a world in which interoperable altchains can be easily created and used, but without unnecessarily fragmenting markets and development. In this paper, we argue that it is possible to simultaneously achieve these seemingly contradictory goals. The core observation is that "Bitcoin" the blockchain is conceptually independent from "bitcoin" the asset: if we had technology to support the movement of assets between blockchains, new systems could be developed which users could adopt by simply reusing the existing bitcoin currency[2].

We refer to such interoperable blockchains as *pegged sidechains*. We will give a precise definition in Section 3, but for now we list the following desired properties for pegged sidechains:

1. Assets which are moved between sidechains should be able to be moved back by whomever their current holder is, and nobody else (including previous holders).

2. Assets should be moved without counterparty risk; that is, there should be no ability for a dishonest party to prevent the transfer occurring.

---

[2]We use bitcoin as an example because its strong network effects make it likely that users will prefer it over other, newer assets. However, any altcoin can be adapted to be usable with sidechains.

3. Transfers should be atomic, *i.e.* happen entirely or not at all. There should not be failure modes that result in loss or allow fraudulent creation of assets.

4. Sidechains should be *firewalled*: a bug in one sidechain enabling creation (or theft) of assets in that chain should not result in creation or theft of assets on any other chain.

5. Blockchain reorganisations[3] should be handled cleanly, even during transfers; any disruption should be localised to the sidechain on which it occurs. In general, sidechains should ideally be *fully independent*, with users providing any necessary data from other chains. Validators of a sidechain should only be required to track another chain if that is an explicit consensus rule of the sidechain itself.

6. Users should not be required to track sidechains that they are not actively using.

An early solution was to "transfer" coins by destroying bitcoins in a publicly recognisable way[4], which would be detected by a new blockchain to allow creation of new coins[Bac13b]. This is a partial solution to the problems listed above, but since it allows only unidirectional transfers between chains, it is not sufficient for our purposes.

Our proposed solution is to transfer assets by providing proofs of possession in the transferring transactions themselves, avoiding the need for nodes to track the sending chain. On a high level, when moving assets from one blockchain to another, we create a transaction on the first blockchain locking the assets, then create a transaction on the second blockchain whose inputs contain a cryptographic proof that the lock was done correctly. These inputs are tagged with an asset type, *e.g.* the genesis hash of its originating blockchain.

We refer to the first blockchain as the *parent chain*, and the second simply as the *sidechain*. In some models, both chains are treated symmetrically, so this terminology should be considered relative. Conceptually, we would like to transfer an asset from the (original) parent chain to a sidechain, possibly onward to another sidechain, and eventually back to the parent chain, preserving the original asset. Generally we can think of the parent chain as being Bitcoin and the sidechain as one of many other blockchains. Of course, sidechain coins could be transferred between sidechains, not just to and from Bitcoin; however, since any coin originally moved from Bitcoin could be moved back, it would nonetheless remain a bitcoin.

This lets us solve the problem of fragmentation described in the previous section, which is good news for cryptocurrency developers who want to focus solely on technical innovation.

Furthermore, because sidechains transfer existing assets from the parent chain rather than creating new ones, sidechains cannot cause unauthorised creation of coins, relying instead on the parent chain to maintain the security and scarcity of its assets[5].

Further still, participants do not need to be as concerned that their holdings are locked in a single experimental altchain, since sidechain coins can be redeemed for an equal number of parent chain coins. This provides an exit strategy, reducing harm from unmaintained software.

---

[3]A *reorganisation*, or *reorg*, occurs locally in clients when a previously accepted chain is overtaken by a competitor chain with more proof of work, causing any blocks on the losing side of the fork to be removed from consensus history.

[4]This is also known as a *one-way peg*, to contrast with *two-way peg*, which we introduce later on.

[5]Of course, sidechains are able to support their own assets, which they would be responsible for maintaining the scarcity of. We mean to emphasise that they can only affect the scarcity of themselves and their child chains.

On the other hand, because sidechains are still blockchains independent of Bitcoin, they are free to experiment with new transaction designs, trust models, economic models, asset issuance semantics, or cryptographic features. We will explore many of the possibilities for sidechains further in Section 5.

An additional benefit to this infrastructure is that making changes to Bitcoin itself becomes much less pressing: rather than orchestrating a fork which all parties need to agree on and implement in tandem, a new "changed Bitcoin" could be created as a sidechain. If, in the medium term, there were wide agreement that the new system was an improvement, it may end up seeing significantly more use than Bitcoin. As there are no changes to parent chain consensus rules, everyone can switch in their own time without any of the risks associated with consensus failure. Then, in the longer term, the success of the changes in the sidechain would provide the needed confidence to change the parent chain, if and when it is deemed necessary to do so.

2   Design rationale

Trustlessness is the property of not relying on trusting external parties for correct operation, typically by enabling all parties to verify on their own that information is correct. For example, in cryptographic signature systems trustlessness is an implicit requirement (signature systems where an attacker can forge signatures would be considered utterly broken). While this is not typical for distributed systems, Bitcoin does provide trustless operation for most parts of its system[6].

A major design goal of pegged sidechains is to minimise additional trust over Bitcoin's model. The hard part is securing transfers of coins between sidechains: the receiving chain must see that the coins on the sending chain were correctly locked. Following Bitcoin's lead, we propose solving this using DMMSes. Although it is possible to use a simple trust-based solution involving fixed signers (see Appendix A) to verify locking of coins, there are important reasons to avoid the introduction of single points of failure:

- Trusting individual signers does not only mean expecting them to behave honestly; they must also never be compromised, never leak secret key material, never be coerced, and never stop participating in the network.

- Because digital assets are long-lived, any trust requirements must be as well. Experience has shown that trust requirements are dangerous expectations even for timespans on the order of months, let alone the generational timespans we expect financial systems to last.

- Digital currencies were unable to gain traction until Bitcoin was able to eliminate single points of failure, and the community is strongly averse to the introduction of such weaknesses. Community mistrust is reinforced by financial events since 2007; public trust in the financial system and other public institutions is likewise at historical lows.

---

[6]This is true for almost all aspects of Bitcoin: a user running a full node will never accept a transaction that is directly or indirectly the result of counterfeiting or spending without proving possession. However, trustless operation is not possible for preventing double spending, as there is no way to distinguish between two conflicting but otherwise valid transactions. Instead of relying on a centralised trusted party or parties to take on this arbitration function like Bitcoin's predecessors, Bitcoin reduces the trust required — but does not remove it — by using a DMMS and economic incentives.

3 Two-way peg

The technical underpinning of pegged sidechains is called the *two-way peg*. In this section we explain the workings thereof, beginning with some definitions.

3.1 Definitions

- A *coin*, or *asset*, is a digital property whose controller can be cryptographically ascertained.

- A *block* is a collection of transactions describing changes in asset control.

- A *blockchain* is a well-ordered collection of blocks, on which all users must (eventually) come to consensus. This determines the history of asset control and provides a computationally unforgeable time ordering for transactions.

- A *reorganisation*, or *reorg*, occurs locally in clients when a previously accepted chain is overtaken by a competitor chain with more proof of work, causing any blocks on the losing side of the fork to be removed from consensus history.

- A *sidechain* is a blockchain that validates data from other blockchains.

- *Two-way peg* refers to the mechanism by which coins are transferred between sidechains and back at a fixed or otherwise deterministic exchange rate.

- A *pegged sidechain* is a sidechain whose assets can be imported from and returned to other chains; that is, a sidechain that supports two-way pegged assets.

- A *simplified payment verification proof* (or *SPV proof*[7]) is a DMMS that an action occurred on a Bitcoin-like proof-of-work blockchain.

  Essentially, an SPV proof is composed of (a) a list of blockheaders demonstrating proof-of-work, and (b) a cryptographic proof that an output was created in one of the blocks in the list. This allows verifiers to check that some amount of work has been committed to the existence of an output. Such a proof may be invalidated by another proof demonstrating the existence of a chain with more work which does not include the block which created the output.

  Using SPV proofs to determine history, implicitly trusting that the longest blockchain is also the longest correct blockchain, is done by so-called *SPV clients* in Bitcoin. Only a dishonest collusion with greater than 50% of the hashpower can persistently fool an SPV client (unless the client is under a long-term Sybil attack, preventing it from seeing the actual longest chain), since the honest hashpower will not contribute work to an invalid chain.

  Optionally, by requiring each blockheader to commit to the blockchain's unspent output set[8], anyone in possession of an SPV proof can determine the state of the chain without needing to

---

[7]Named after the section 'Simplified Payment Verification' in [Nak09]

[8]In Bitcoin, only the set of *unspent transaction outputs (UTXO's)* is needed to determine the status of all coins. By constructing a Merkle tree[Mer88], we can commit to every element of the UTXO set using only a single hash, minimising the blockheader space used.

"replay" every block. (In Bitcoin, full verifiers need to do this when they first start tracking the blockchain.)

As we will discuss in Appendix B, by including some additional data in Bitcoin's block structure, we can produce smaller proofs than a full list of headers, which will improve scalability. Still, these proofs will be much larger than ordinary Bitcoin transactions. Fortunately, they are not necessary for most transfers: holders of coins on each chain may exchange them directly using atomic swaps[Nol13], as described in Appendix C.

3.2   Symmetric two-way peg

We can use these ideas to *SPV peg* one sidechain to another. This works as follows: to transfer parent chain coins into sidechain coins, the parent chain coins are sent to a special output on the parent chain that can only be unlocked by an *SPV proof* of possession on the sidechain. To synchronise the two chains, we need to define two waiting periods:

1. The *confirmation period* of a transfer between sidechains is a duration for which a coin must be locked on the parent chain before it can be transferred to the sidechain. The purpose of this confirmation period is to allow for sufficient work to be created such that a denial of service attack in the next waiting period becomes more difficult. A typical confirmation period would be on the order of a day or two.

   After creating the special output on the parent chain, the user waits out the confirmation period, then creates a transaction on the sidechain referencing this output, providing an SPV proof that it was created and buried under sufficient work on the the parent chain.

   The confirmation period is a per-sidechain security parameter, which trades cross-chain transfer speed for security.

2. The user must then wait for the *contest period*. This is a duration in which a newly-transferred coin may not be spent on the sidechain. The purpose of a contest period is to prevent double-spending by transferring previously-locked coins during a reorganisation. If at any point during this delay, a new proof is published containing a chain with more aggregate work which does not include the block in which the lock output was created, the conversion is retroactively invalidated. We call this a *reorganisation proof*.

   All users of the sidechain have an incentive to produce reorganisation proofs if possible, as the consequence of a bad proof being admitted is a dilution in the value of all coins.

   A typical contest period would also be on the order of a day or two. To avoid these delays, users will likely use atomic swaps (described in Appendix C) for most transfers, as long as a liquid market is available.

While locked on the parent chain, the coin can be freely transferred within the sidechain without further interaction with the parent chain. However, it retains its identity as a parent chain coin, and can only be transferred back to the same chain that it came from.

When a user wants to transfer coins from the sidechain back to the parent chain, they do the same thing as the original transfer: send the coins on the sidechain to an SPV-locked output, produce a

9

Parent Chain   Sidechain
            ⋮   ⋮
Send to SPV-locked output
Wait out confirmation period
                                          SPV Proof
                            ─────────────────────────────────▷
                                                                    Wait out contest period
                                                                              ⋮
                                                                    (Intra-chain transfers)
                                                                              ⋮
                                                                    Send to SPV-locked output
                                                                    Wait out confirmation period
                                          SPV proof
                            ◁─────────────────────────────────
Contest period begins
                                  SPV reorganisation Proof
                            ◁─────────────────────────────────
Contest period ends (failed)
                                       New SPV proof
                            ◁─────────────────────────────────
Wait out contest period
            ⋮
(Intra-chain transfers)
            ⋮   ⋮
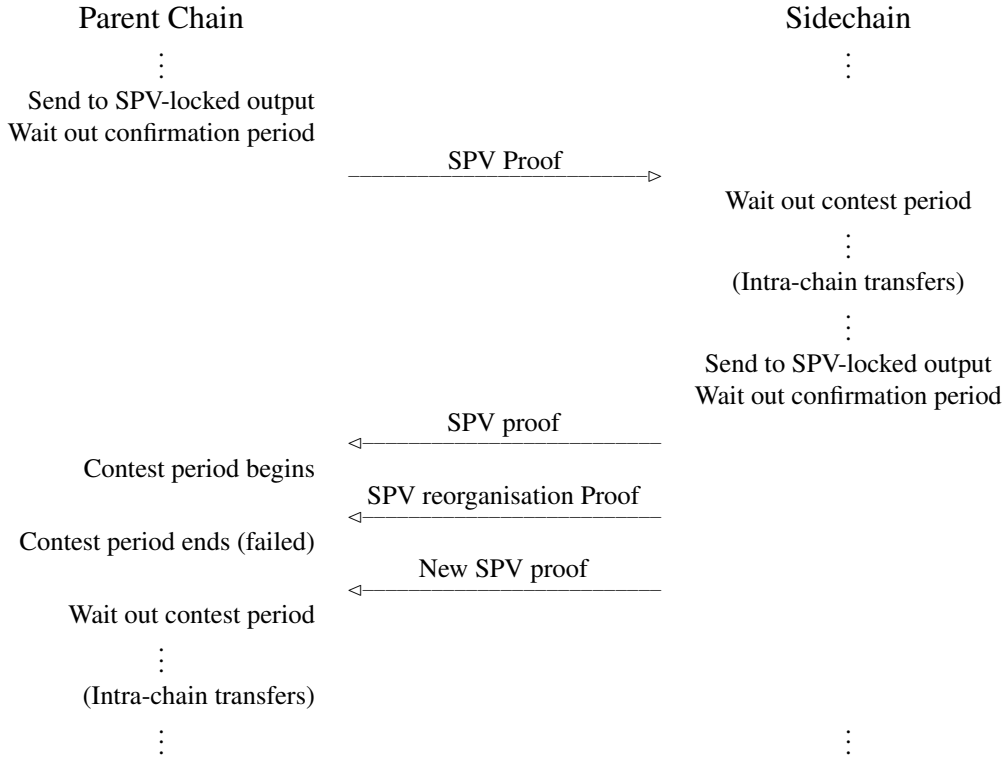
Figure 1: Example two-way peg protocol.

sufficient SPV proof that this was done, and use the proof to unlock a number of previously-locked outputs with equal denomination on the parent chain. The entire transfer process is demonstrated in Figure 1.

Since pegged sidechains may carry assets from many chains, and cannot make assumptions about the security of these chains, it is important that different assets are not interchangeable (except by an explicit trade). Otherwise a malicious user may execute a theft by creating a worthless chain with a worthless asset, move such an asset to a sidechain, and exchange it for something else. To combat this, sidechains must effectively treat assets from separate parent chains as separate asset types.

In summary, we propose to make the parent chain and sidechains do SPV validation of data on each other. Since the parent chain clients cannot be expected to observe every sidechain, users import proofs of work from the sidechain into the parent chain in order to prove possession. In a symmetric two-way peg, the converse is also true.

To use Bitcoin as the parent chain, an extension to script which can recognise and validate such SPV proofs would be required. At the very least, such proofs would need to be made compact enough to fit in a Bitcoin transaction. However, this is just a soft-forking change[9], without effect on transactions which do not use the new features.

─────────────────────────

[9]A soft-forking change is a change which only imposes further restrictions on what is valid inside the chain. See Section 4.4 for more information.

3.3 Asymmetric two-way peg

The previous section was titled "*Symmetric* Two-Way Peg" because the transfer mechanisms from parent chain to sidechain and back were the same: both had SPV security[10].

An alternate scheme is an *asymmetric* two-way peg: here users of the sidechain are full validators of the parent chain, and transfers from parent chain to sidechain do not require SPV proofs, since all validators are aware of the state of the parent chain. On the other hand, the parent chain is still unaware of the sidechain, so SPV proofs are required to transfer back.

This gives a boost in security, since now even a 51% attacker cannot falsely move coins from the parent chain to the sidechain. However, it comes at the expense of forcing sidechain validators to track the parent chain, and also implies that reorganisations on the parent chain may cause reorganisations on the sidechain. We do not explore this possibility in detail here, as issues surrounding reorganisations result in a significant expansion in complexity.

4 Drawbacks

While sidechains provide solutions to many problems in the cryptocurrency space, and create countless opportunities for innovation to Bitcoin, they are not without their drawbacks. In this section we review a few potential problems, along with solutions or workarounds.

4.1 Complexity

Sidechains introduce additional complexity on several levels.

On the network level, we have multiple independent unsynchronised blockchains supporting transfers between each other. They must support transaction scripts which can be invalidated by a later reorganisation proof. We also need software which can automatically detect misbehaviour and produce and publish such proofs.

On the level of assets, we no longer have a simple "one chain, one asset" maxim; individual chains may support arbitrarily many assets, even ones that did not exist when the chain was first created. Each of these assets is labelled with the chain it was transferred from to ensure that their transfers can be unwound correctly.

Enabling the blockchain infrastructure to handle advanced features isn't sufficient: user interfaces for managing wallets will need to be reconsidered. Currently in the altcoin world, each chain has its own wallet which supports transactions of that chain's coin. These will need to adapt to support multiple chains (with potentially different feature sets) and transfers of assets between chains. Of course, there is always the option of not using some functionality when the interface would be too complex.

---

[10]This means using the DMMS not only for determining the order of transactions, but also their validity. In other words, this means trusting miners to not create invalid blocks.

4.2 Fraudulent transfers

Reorganisations of arbitrary depth are in principle possible, which could allow an attacker to completely transfer coins between sidechains before causing a reorganisation longer than the contest period on the sending chain to undo its half of the transfer. The result would be an imbalance between the number of coins on the recipient chain and the amount of locked output value backing them on the sending chain. If the attacker is allowed to return the transferred coins to the original chain, he would increase the number of coins in his possession at the expense of other users of the sidechain.

Before discussing how to handle this, we observe that this risk can be made arbitrarily small by simply increasing the contest period for transfers. Better, the duration of the contest period could be made a function of the relative hashpower of the two chains: the recipient chain might only unlock coins given an SPV proof of one day's worth of *its own* proof-of-work, which might correspond to several days of the sending chain's proof-of-work. Security parameters like these are properties of the particular sidechain and can be optimised for each sidechain's application.

Regardless of how unlikely this event is, it is important that the sidechain not respond with catastrophic failure. It is possible to create an SPV proof witnessing such an event, and sidechains may accept such proofs. They may be designed to react in one of many possible ways:

- No reaction. The result is that the sidechain is a "fractional reserve" of the assets it is storing from other chains. This may be acceptable for tiny amounts which are believed to be less than the number of lost sidechain coins, or if an insurer promises to make good on missing assets. However, beyond some threshold, a "bank run" of withdrawals from the sidechain is likely, which would leave somebody holding the bag in the end. Indirect damage could include widespread loss of faith in sidechains, and the expense to the parent chain to process a sudden rush of transactions.

- The peg and all dependent transactions could be reversed. However, as coins tend to diffuse and histories intermingle, the effect of such a reversal could be catastrophic after even short periods of time. It also limits fungibility, as recipients would prefer coins with "clean" histories (no recent pegs). We expect that such a loss of fungibility might have disastrous consequences.

- The amount of all coins could be reduced, while leaving the exchange rate intact. Now users who transferred coins to the sidechain prior to the attack are disadvantaged relative to new ones. Reducing the exchange rate for sidechain coins would be equivalent.

Many variations on these reactions are possible: for example, temporarily decreasing the exchange rate so those who "make a run" on the sidechain cover the loss of those who don't.

4.3 Risk of centralisation of mining

An important concern is whether the introduction of sidechains with mining fees places resource pressure on miners, creating Bitcoin centralisation risks.

12

Because miners receive compensation from the block subsidy and fees of each chain they provide work for, it is in their economic interest to switch between providing DMMSes for different similarly-valued blockchains following changes in difficulty and movements in market value.

One response is that some blockchains have tweaked their blockheader definition such that it includes a part of Bitcoin's DMMS, thus enabling miners to provide a single DMMS that commits to Bitcoin as well as one or more other blockchains — this is called *merged mining*. Since merged mining enables re-use of work for multiple blockchains, miners are able to claim compensation from each blockchain that they provide DMMSes for.

As miners provide work for more blockchains, more resources are needed to track and validate them all. Miners that provide work for a subset of blockchains are compensated less than those which provide work for every possible blockchain. Smaller-scale miners may be unable to afford the full costs to mine every blockchain, and could thus be put at a disadvantage compared to larger, established miners who are able to claim greater compensation from a larger set of blockchains.

We note however that it is possible for miners to delegate validation and transaction selection of any subset of the blockchains that they provide work for. Choosing to delegate authority enables miners to avoid almost all of the additional resource requirements, or provide work for blockchains that they are still in the process of validating. However such delegation comes at the cost of centralising validation and transaction selection for the blockchain, even if the work generation itself remains distributed. Miners might also choose instead to not provide work for blockchains that they are still in the process of validating, thus voluntarily giving up some compensation in exchange for increased validation decentralisation.

4.4   Risk of soft-fork

In Bitcoin, a *soft-fork* is an addition to the Bitcoin protocol made backwards compatible by being designed to strictly reduce the set of valid transactions or blocks. A soft-fork can be implemented with merely a supermajority of the mining computational power participating, rather than all full nodes. However, participants' security with respect to the soft-forked features is only SPV-level until they upgrade. Soft-forks have been used several times to deploy new features and fix security issues in Bitcoin (see [And12b]).

A two-way peg, implemented as described in this paper, has only SPV security and therefore has greater short-term dependence on miner honesty than Bitcoin does (see the attack described in Section 4.2). However, a two-way peg can be boosted to security absolutely equal to Bitcoin's if all full nodes on both systems inspect each other's chain and demand mutual validity as a soft-forking rule.

A negative consequence of this would be loss of isolation of any soft-fork-required sidechain. Since isolation was one of the goals of using pegged sidechains, this result would be undesirable unless a sidechain was almost universally used. Absent pegged sidechains, however, the next alternative would be to deploy individual changes as hard- or soft-forks in Bitcoin directly. This is even more abrupt, and provides no real mechanism for the new facility to prove its maturity and demand before risking Bitcoin's consensus on it.

5 Applications

With the technical underpinnings out of the way, in this section we explore user-facing applications of sidechains, which effectively extend Bitcoin to do things that it cannot today.

5.1 Altchain experiments

The first application, already mentioned many times, is simply creating altchains with coins that derive their scarcity and supply from Bitcoin. By using a sidechain which carries bitcoins rather than a completely new currency, one can avoid the thorny problems of initial distribution and market vulnerability, as well as barriers to adoption for new users, who no longer need to locate a trustworthy marketplace or invest in mining hardware to obtain altcoin assets.

5.1.1 Technical experimentation

Because sidechains are technically still fully-independent chains, they are able to change features of Bitcoin such as block structure or transaction chaining. Some examples of such features are:

- By fixing undesired transaction malleability[11] — which can only be fixed partially in Bitcoin [Wui14] — protocols which involve chains of unconfirmed transactions can be executed safely. Transaction malleability is a problem in Bitcoin which allows arbitrary users to tweak transaction data in a way that breaks any later transactions which depend on them, even though the actual content of the transaction is unchanged. An example of a protocol broken by transaction malleability is *probabilistic payments*[Cal12].

- Improved payer privacy, *e.g.* the ring signature scheme used by Monero, can reduce the systemic risk of the transactions of particular parties being censored, protecting the fungibility of the cryptocurrency. Improvements to this have been suggested by Maxwell and Poelstra [MP14, Poe14b] and Back[Bac13a], which would allow for even greater privacy. Today, ring signatures can be used with Monero coins, but not bitcoins; sidechains would avoid this exclusivity.

- Script extensions (for example, to efficiently support coloured coins[jl213]) have been proposed for Bitcoin. Since such extensions are usable only by a small subset of users, but all users would need to deal with the increased complexity and risk of subtle interactions, these extensions have not been accepted into Bitcoin.

  Other suggested script extensions include support for new cryptographic primitives. For example, Lamport signatures[Lam79], while large, are secure against quantum computers.

- Many ideas for extending Bitcoin in incompatible ways are described at [Max14] and at `http://www.bitcoin.ninja`.

---

[11]Note that some forms of malleability are desired (*i.e.* the types provided by SIGHASH flags other than SIGHASH_ALL).

14

Since changes like these affect only the transfer of coins, rather than their creation, there is no need for them to require a separate currency. With sidechains, users can safely and temporarily experiment with them. This encourages adoption for the sidechain, and is less risky for participants relative to using an entirely separate altcoin.

5.1.2 Economic experimentation

Bitcoin's reward structure assigns new coins to miners. This effectively inflates the currency but it winds down over time according to a step-wise schedule. Using this inflation to subsidise mining has been a successful complement to transaction fees to secure the network.

An alternate mechanism for achieving block rewards on the sidechain is demurrage, an idea pioneered for digital currency by Freicoin (`http://freico.in`). In a demurring cryptocurrency, all unspent outputs lose value over time, with the lost value being recollected by miners. This keeps the currency supply stable while still rewarding miners. It may be better aligned with user interests than inflation because loss to demurrage is enacted uniformly everywhere and instantaneously, unlike inflation; it also mitigates the possibility of long-unspent "lost" coins being reanimated at their current valuation and shocking the economy, which is a perceived risk in Bitcoin. Demurrage creates incentives to increase monetary velocity and lower interest rates, which are considered (*e.g.* by Freicoin advocates and other supporters of Silvio Gesell's theory of interest[Ges16]) to be socially beneficial. In pegged sidechains, demurrage allows miners to be paid in existing already-valued currency.

Other economic changes include required miner fees, transaction reversibility, outputs which are simply deleted once they reach a certain age, or inflation/demurrage rates pegged to events outside of the sidechain. All of these changes are difficult to do safely, but the ease of creation and reduced risk of sidechains provide the necessary environment for them to be viable.

5.2 Issued assets

To this point, we have mostly been thinking about sidechains which do not need their own native currency: all coins on the sidechain are initially locked, until they are activated by a transfer from some other sidechain. However, it is possible for sidechains to produce their own tokens, or *issued assets*, which carry their own semantics. These can be transferred to other sidechains and traded for other assets and currencies, all without trusting a central party, even if a trusted party is needed for future redemption.

Issued asset chains have many applications, including traditional financial instruments such as shares, bonds, vouchers, and IOUs. This allows external protocols to delegate ownership and transfer tracking to the sidechain on which the ownership shares were issued. Issued asset chains may also support more innovative instruments such as smart property.

These technologies can also be used in *complementary currencies*[Lie01]. Examples of complementary currencies include community currencies, which are designed to preferentially boost local businesses; business barter associations, which support social programs like education or elderly care; and limited-purpose tokens which are used within organisations such as massive

multiplayer games, loyalty programs, and online communities[12].

A suitably extended scripting system and an asset-aware transaction format would allow the creation of useful transactions from well-audited components, such as the merger of a bid and an ask to form an exchange transaction, enabling the creation of completely trustless peer-to-peer marketplaces for asset exchange and more complex contracts such as trustless options[FT13]. These contracts could, for example, help reduce the volatility of bitcoin itself.

6 Future directions

6.1 Hashpower attack resistance

The main thrust of this paper surrounds two-way peg using SPV proofs, which are forgeable by a 51%-majority and blockable by however much hashpower is needed to build a sufficiently-long proof during the transfer's contest period. (There is a tradeoff on this latter point — if 33% hashpower can block a proof, then 67% is needed to successfully use a false one, and so on.)

Some other ideas worth exploring in sidechains are:

- **Assurance contracts.** The sidechain's transaction fees are withheld from miners unless their hashpower is at least, say, 66% of that of Bitcoin. These sorts of contracts are easy for a cryptocurrency to implement, if they are designed in from the start, and serve to increase the cost of blocking transfers.

- **Time-shifted fees.** Miners receive part of their fees in a block far in the future (or spread across many blocks) so that they have incentive to keep the chain operational.

  This may incentivise miners to simply receive fees out-of-band, avoiding the need to wait for future in-chain rewards. A variation on this scheme is for miners to receive a token enabling them to mine a low-difficulty block far in the future; this has the same effect, but directly incentivises its recipient to mine the chain.

- **Demurrage.** Block subsidies can be given to miners through demurrage to incentivise honest mining. Since only as much can be transferred to Bitcoin or another sidechain as was transferred out, this fund reallocation would be localised to the sidechain in which it occurs.

- **Subsidy.** A sidechain could also issue its own separate native currency as reward, effectively forming an altcoin. However, these coins would have a free-floating value and as a result would not solve the volatility and market fragmentation issues with altcoins.

- **Co-signed SPV proofs.** Introducing signers who must sign off on valid SPV proofs, watching for false proofs. This results in a direct tradeoff between centralisation and security against a high-hashpower attack. There is a wide spectrum of trade-offs available in this area: signers may be required only for high-value transfers; they may be required only when sidechain hashpower is too small a percentage of Bitcoin's; etc. Further discussion about the usefulness of this kind of trade-off is covered in Appendix A.

---

[12]For more information on complementary currencies, see `http://www.complementarycurrency.org/`

- **SNARKs.** An exciting recent development in academic cryptography has been the invention of SNARKs [BSCG⁺13]. SNARKs are space-efficient, quickly verifiable zero-knowledge cryptographic proofs that some computation was done. However, their use is currently inhibited because the proofs for most programs are too slow to generate on today's computers, and the existing constructions require a *trusted setup*, meaning that the creator of the system is able to create false proofs.

A futuristic idea for a low-value or experimental sidechain is to invoke a trusted authority, whose only job is to execute a trusted setup for a SNARK scheme. Then blocks could be constructed which prove their changes to the unspent-output set, but do so in zero-knowledge in the actual transactions. They could even commit to the full verification of all previous blocks, allowing new users to get up to speed by verifying only the single latest block. These proofs could also replace the DMMSes used to move coins from another chain by proving that the sending chain is valid according to some rules previously defined.

7   Acknowledgements

Appendix A   Federated peg

One of the challenges in deploying pegged sidechains is that Bitcoin script is currently not expressive enough to encode the verification rules for an SPV proof. The required expressiveness could be added in a safe, compatible, and highly compartmentalised way (*e.g.*, by converting a no-op instruction into an `OP_SIDECHAINPROOFVERIFY` in a soft-fork). However, the difficulty of building consensus for and deploying even simple new features is non-trivial. Recall these difficulties were part of the motivation for pegged sidechains to begin with. What we want is a way to try out future script capabilities for Bitcoin without deploying them everywhere.

Fortunately, by adopting some additional security assumptions at the expense of the low trust design objective, it is possible to do an initial deployment in a completely permissionless way. The key observation is that any enhancement to Bitcoin Script can be implemented externally by having a trusted federation of mutually distrusting *functionaries*[13] evaluate the script and accept by signing for an ordinary multisignature script. That is, the functionaries act as a *protocol adaptor* by evaluating the same rules we would have wanted Bitcoin to evaluate, but cannot for lack of script enhancements. Using this we can achieve a *federated peg*.

---

[13] From Wiktionary (`https://en.wiktionary.org/wiki/functionary`), a functionary is

> A person ...who holds limited authority and primarily serves to carry out a simple function for which discretion is not required.

We use this term to emphasise that while functionaries have the physical power to disrupt transfers between sidechains, their correct operation is purely mechanical.

This approach is very similar to the approach of creating a multi-signature off-chain transaction system, but the required server-to-server consensus process is provided by simply observing the blockchains in question. The result is a deterministic, highly-auditable process which simplifies the selection and supervision of functionaries. Because of these similarities, many of the techniques used to improve security and confidence in off-chain payment systems can be employed for federated pegs. For example: functionaries can be geographically diverse, bonded via escrowed coins or expensive-to-create coercion-resistant pseudonymous identities, implemented on remote-attesting tamper-resistant hardware, and so on[Tod13]. For small-scale uses, owners of coins in the system can themselves act as the functionaries, thus avoiding third party trust.

Once sidechains with a federated peg are in use, the addition of SPV verification to Bitcoin script can be seen as merely a security upgrade to reduce the trust required in the system. Existing sidechains could simply migrate their coins to the new verification system. This approach also opens additional security options: the DMMS provided by mining is not very secure for small systems, while the trust of the federation is riskier for large systems. A sidechain could adaptively use both of these approaches in parallel, or even switch based on apparent hashrate.

Consider the example of a sidechain using a 3 of 5 federation of functionaries to implement a two-way peg with Bitcoin. The federation has secp256k1 public points (public keys) P_1, P_2, P_3, P_4, and P_5 and a redeemscript template 3 x x x x x 5 OP_CHECKMULTISIG known to all participants in the sidechain. To send coins to a ScriptPubKey *SPK*, a user who wants the coins to become available on a sidechain using the federated peg computes a cross-chain P2SH[And12a] address by the following key derivation scheme:

Algorithm 1 GenerateCrossChainAddress

Input: A target ScriptPubKey *SPK* which will receive the coins in the other chain
Input: A list $\{P_i\}_{i=1}^n$ of the functionaries' public points
Input: A redeemScript *template* describing the functionary requirements
Output: A P2SH address
Output: Nonce used for this instance
 1: $nonce \leftarrow$ random_128bit()
 2: **for** $i \leftarrow [1,n]$ **do**
 3:     $Tweak_i \leftarrow$ HMAC-SHA256($key = P_i$, $data = nonce||SPK$)
 4:     **if** $Tweak_i >=$ secp256k1_order **then**
 5:         Go back to start.
 6:     **end if**
 7:     $PCC_i = P_i + G \times Tweak_i$
 8: **end for**
 9: $address \leftarrow$ P2SH_Multisig($template, keys = PCC$))

This derivation scheme is based on the same homomorphic technique [Max11] used in BIP32 to allow third parties to derive publicly unlinkable addresses. It is the same underlying construction as a pay-to-contract transaction [GH12]. After generating the address, coins can be paid to it, and the user can later receive the resulting coins on the sidechain by providing the functionaries with the nonce, ScriptPubKey, and an SPV proof to help them locate the payment in the blockchain. In order to aid third-party verification of the sidechain, these values could be included in the sidechain itself. Because the transfer is made by paying to a standard P2SH address and can pay to any

ScriptPubKey, all Bitcoin services which can pay to a multisignature address will immediately be able to pay into, or receive payments from, a user using a federated sidechain.

The federated peg approach necessarily compromises on trust, but requires no changes to Bitcoin — only the participants need to agree to use it and only the participants take the costs or risks of using it. Further, if someone wanted to prevent other people from using a sidechain they could not do so: if the federated peg is used privately in a closed community, its use can be made undetectable and uncensorable. This approach allows rapid deployment and experimentation and will allow the community to gain confidence in pegged sidechains before adopting any changes to the Bitcoin protocol.

Appendix B   Efficient SPV proofs

In order to transfer coins from a sidechain back to Bitcoin, we need to embed proofs that sidechain coins were locked in the Bitcoin blockchain. These proofs should contain (a) a record that an output was created in the sidechain, and (b) a DMMS proving sufficient work on top of this output. Because Bitcoin's blockchain is shared and validated by all of its participants, these proofs must not impose much burden on the network. Outputs can be easily recorded compactly, but it is not obvious that the DMMS can be.

Compact SPV Security.   The confidence in an SPV proof can be justified by modelling an attacker and the honest network as random processes [MLJ14]. These random processes have a useful statistical property: while each hash must be less than its target value to be valid, half the time it will be less than half the target; a third of the time it will be less than a third the target; a quarter of the time less than a quarter the target; and so on. While the hash value itself does not change the amount of work a block is counted as, the presence of lower-than-necessary hashes is in fact statistical evidence of more work done in the chain[Mil12]. We can exploit this fact to prove equal amounts of work with only a few block headers[Fri14]. It should therefore be possible to greatly compress a list of headers while still proving the same amount of work. We refer to such a compressed list as a *compact SPV proof* or *compressed DMMS*.

However, while the expected work required to produce a fraudulent compact SPV proof is the same as that for a non-compact one, a forger's probability of success no longer decays exponentially with the amount of work proven: a weak opportunistic attacker has a much higher probability of succeeding "by chance"; *i.e.*, by finding low hashes early. To illustrate this, suppose such an attacker has 10% of the network's hashrate, and is trying to create an SPV proof of 1000 blocks before the network has produced this many. Following the formula in [Nak09] we see that his likelihood of success is

$$1 - \sum_{k=0}^{1000} \frac{100^k e^{-100}}{k!} (1 - (0.1)^{1000-k}) \approx 10^{-196}$$

To contrast, the same attacker in the same time can produce a single block proving 1000 blocks' worth of work with probability roughly 10%, a much higher number.

A detailed analysis of this problem and its possible solutions is out of scope for this document.

For now we will describe an implementation of compact SPV proofs, along with some potential solutions to block this sort of attack while still obtaining significant proof compaction.

Note that we are assuming a constant difficulty. We observe that Bitcoin's difficulty, while non-constant, changes slowly enough to be resistant to known attacks[Bah13]. We therefore expect that corrections which take into account the adjusting difficulty can be made.

Implementation. The inspiration for compact SPV proofs is the *skiplist* [Pug90], a probabilistic data structure which provides log-complexity search without requiring rebalancing (which is good because an append-only structure such as a blockchain cannot be rebalanced).

We require a change to Bitcoin so that rather than each blockheader committing only to the header before it, it commits to every one of its ancestors. These commitments can be stored in a Merkle tree for space efficiency: by including only a root hash in each block, we obtain a commitment to every element in the tree. Second, when extracting SPV proofs, provers are allowed to use these commitments to jump back to a block more than one link back in the chain, *provided* the work actually proven by the header exceeds the total target work proven by only following direct predecessor links. The result is a short DMMS which proves just as much work as the original blockchain.

How much smaller is this? Suppose we are trying to produce an SPV proof of an entire blockchain of height N. Assume for simplicity that difficulty is constant for the chain; *i.e.*, every block target is the same. Consider the probability of finding a large enough proof to skip all the way back to the genesis within x blocks; that is, between block $N - x$ and block N. This is one minus the probability we *don't*, or

$$1 - \prod_{i=1}^{x} \frac{N-i}{N-i+1} = 1 - \frac{N-x}{N} = \frac{x}{N}$$

The expected number of blocks needed to scan back before skipping the remainder of the chain is thus

$$\sum_{x=1}^{N} \frac{x}{N} = \frac{N+1}{2}$$

Therefore if we want to skip the entire remaining chain in one jump, we expect to search only halfway; by the same argument we expect to skip this half after only a quarter, this quarter after only an eighth, and so on. The result is that the expected total proof length is logarithmic in the original length of the chain.

For a million-block chain, the expected proof size for the entire chain is only $\log_2 1000000 \approx 20$ headers. This brings the DMMS size down into the tens-of-kilobytes range.

However, as observed above, if an attacker is able to produce compact proofs in which only the revealed headers are actually mined, he is able to do so with non-negligible probability in the total work being proven. One such strategy is for the attacker to produce invalid blocks in which every backlink points to the most recent block. Then when extracting a compact proof, the attacker simply follows the highest-weighted link every time.

We can adapt our scheme to prevent this in one of several ways:

- By limiting the maximum skip size, we return to Bitcoin's property that the likelihood of a probabilistic attack decays exponentially with the amount of work being proven. The expected proof size is smaller than a full list of headers by a constant (proportional to the maximum skip size) factor.

- By using a maximum skip size which increases with the amount of work being proven it is possible to get sublinear proof sizes, at the cost of subexponential decay in the probability of attack success. This gives greater space savings while still forcing a probabilistic attacker's likelihood of success low enough to be considered negligible.

- Interactive approaches or a cut-and-choose mechanism may allow compact proofs with only a small security reduction. For example, provers might be required to reveal random committed blockheaders (and their connection to the chain), using some part of the proof as a random seed. This reduces the probability of attack while only increasing proof size by a constant factor.

If we expect many transfers per sidechain, we can maintain a special output in the parent chain which tracks the sidechain's tip. This output is moved by separate SPV proofs (which may be compacted in one of the above ways), with the result that the parent chain is aware of a recent sidechain's tip at all times.

Then transfer proofs would be required to always end at this tip, which can be verified with only a single output lookup. This guarantees verifiers that there are no "missing links" in the transfer proofs, so they may be logarithmic in size without increased risk of forgery.

This makes the total cost to the parent chain proportional to the number of sidechains and their length; without this output, the total cost is also proportional to the number of inter-chain transfers.

This discussion is not exhaustive; optimising these tradeoffs and formalising the security guarantees is out of scope for this paper and the topic of ongoing work.

Appendix C   Atomic swaps

Once a sidechain is operational, it is possible for users to exchange coins atomically between chains, without using the peg. In fact, this is possible with altcoins today, though the independent prices make it harder to organise. This is important, because as we have seen, direct use of the peg requires fairly large transactions (with correspondingly large fees) and long wait periods. To contrast, atomic swaps can be done using only two transactions on each network, each of size similar to ordinary pay-to-address transactions.

One such scheme, due to Tier Nolan[Nol13], works as follows.

Suppose we have two parties, A and B, who hold coins on different blockchains. Suppose also that they each have addresses pk_A and pk_B on the other's chain, and that A has a secret number a. Then A can exchange coins for B's as follows:

1. On one chain, A creates a transaction moving coins to an output O_1 which can only be redeemed with (a) a revealing of a and B's signature, or (b) both A and B's signatures. A does not yet broadcast this.

A creates a second transaction returning the coins from O_1 to *A*, with a locktime[14] of 48 hours. *A* passes this transaction to *B* to be signed.

Once *B* signs the locked refund transaction, *A* may safely broadcast the transaction moving coins to O_1, and does so.

2. Similarly, *B* creates a transaction moving coins to an output O_2 on the other chain, which can only be redeemed by (a) a revealing of *a* and *A*'s signature, or (b) both *A* and *B*'s signatures. *B* does not yet broadcast this.

   B creates a second transaction returning the coins from O_2 to *B*, with a locktime of 24 hours. *B* passes this transaction to *A* to be signed.

   Once *A* signs the locked refund transaction, *B* may safely broadcast the transaction moving his coins to O_2, and does so.

3. Since *A* knows *a*, *A* is able to spend the coins in O_2, and does so, taking possession of *B*'s coins.

   As soon as *A* does so, *a* is revealed and *B* becomes able to spend the coins in O_1, and does so, taking possession of *A*'s coins.

---

[14]In Bitcoin, a transaction's *locktime* prevents it from being included in the blockchain until some timeout has expired. This is useful for creating refunds in interactive protocols which can only be redeemed if the protocol times out.

References

[AJK05]   J. Aspnes, C. Jackson, and A. Krishnamurthy, *Exposing computationally-challenged Byzantine impostors*, Tech. Report YALEU/DCS/TR-1332, Yale University, 2005, `http://www.cs.yale.edu/homes/aspnes/papers/tr1332.pdf`.

[And12a]   G. Andresen, *BIP16: Pay to script hash*, Bitcoin Improvement Proposal, 2012, `https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki`.

[And12b]   _____, *BIP34: Block v2, height in coinbase*, Bitcoin Improvement Proposal, 2012, `https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki`.

[Bac02]   A. Back, *Hashcash — a denial of service counter-measure*, 2002, `http://hashcash.org/papers/hashcash.pdf`.

[Bac13a]   _____, *bitcoins with homomorphic value (validatable but encrypted)*, 2013, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=305791.0`.

[Bac13b]   _____, *Re: [Bitcoin-development] is there a way to do bitcoin-staging?*, 2013, Mailing list post, `http://sourceforge.net/p/bitcoin/mailman/message/31519067/`.

[Bah13]   L. Bahack, *Theoretical Bitcoin attacks with less than half of the computational power (draft)*, arXiv preprint arXiv:1312.7013 (2013).

[BSCG+13]   E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, *SNARKs for C: Verifying program executions succinctly and in zero knowledge*, Cryptology ePrint Archive, Report 2013/507, 2013, `http://eprint.iacr.org/2013/507`.

[Cal12]   M. Caldwell, *Sustainable nanopayment idea: Probabilistic payments*, 2012, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=62558.0`.

[Cha83]   D. Chaum, *Blind signatures for untraceable payments*, Advances in Cryptology Proceedings of Crypto **82** (1983), no. 3, 199–203.

[Fri14]   M. Friedenbach, *[Bitcoin-development] compact SPV proofs via block header commitments*, 2014, Mailing list post, `http://sourceforge.net/p/bitcoin/mailman/message/32111357/`.

[FT13]   M. Friedenbach and J. Timón, *Freimarkets: extending bitcoin protocol with user-specified bearer instruments, peer-to-peer exchange, off-chain accounting, auctions, derivatives and transitive transactions*, 2013, `http://freico.in/docs/freimarkets-v0.0.1.pdf`.

[Ges16]   Silvio Gesell, *The natural economic order*, Peter Owen Ltd. 1958., London, 1916, `https://archive.org/details/TheNaturalEconomicOrder`.

[GH12]   I. Gerhardt and T. Hanke, *Homomorphic payment addresses and the pay-to-contract protocol*, CoRR **abs/1212.3257** (2012).

[Gri99]   Ian Grigg, Email correspondence, 1999, `http://cryptome.org/jya/digicrash.htm`.

[jl213]   jl2012, *OP_CHECKCOLORVERIFY: soft-fork for native color coin support*, 2013, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=253385.0`.

[Lam79]   L. Lamport, *Constructing digital signatures from a one-way function*, Tech. Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[Lie01]   B. Lietear, *The future of money*, Random House, London, January 2001.

[Max11]   G. Maxwell, *Deterministic wallets*, 2011, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=19137.0`.

[Max14]   _____, *User:gmaxwell/alt ideas*, `https://en.bitcoin.it/wiki/User:Gmaxwell/alt_ideas`. Retrieved on 2014-10-09., 2014.

[Mer88]   R.C. Merkle, *A digital signature based on a conventional encryption function*, Lecture Notes in Computer Science, vol. 293, 1988, p. 369.

[Mil12]   A. Miller, *The high-value-hash highway*, 2012, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=98986.0`.

[MLJ14]   A. Miller and J. J. LaViola Jr, *Anonymous Byzantine consensus from moderately-hard puzzles: A model for Bitcoin*, Tech. Report CS-TR-14-01, UCF, April 2014.

[Mou13]   Y. M. Mouton, *Increasing anonymity in Bitcoin ... (possible alternative to Zerocoin?)*, 2013, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=290971`.

[MP14]   G. Maxwell and A. Poelstra, *Output distribution obfuscation*, `https://download.wpsoftware.net/bitcoin/wizardry/brs-arbitrary-output-sizes.txt`, 2014.

[Nak09]   S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2009, `https://www.bitcoin.org/bitcoin.pdf`.

[Nol13]   T. Nolan, *Re: Alt chains and atomic transfers*, `https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949`, 2013.

[Poe14a]   A. Poelstra, *ASICs and decentralization FAQ*, 2014, `https://download.wpsoftware.net/bitcoin/asic-faq.pdf`.

[Poe14b]   _____, *Is there any _true_ anonymous cryptocurrencies?*, Bitcoin.SE, 2014, `http://bitcoin.stackexchange.com/a/29473`.

[Poe14c]   _____, *A treatise on altcoins*, 2014, Unfinished, `https://download.wpsoftware.net/bitcoin/alts.pdf`.

[Pug90]   W. Pugh, *Skip lists: A probabilistic alternative to balanced trees*, Communications of the ACM **33** (1990), no. 6, 668, `ftp://ftp.cs.umd.edu/pub/skipLists/skiplists.pdf`.

[Sza97]   N. Szabo, *The idea of smart contracts*, 1997, `http://szabo.best.vwh.net/idea.html`.

[Tod13]   P. Todd, *Fidelity-bonded banks: decentralized, auditable, private, off-chain payments*, 2013, BitcoinTalk post, `https://bitcointalk.org/index.php?topic=146307.0`.

[vS13]   N. van Saberhagen, *Cryptonote v 2.0*, `https://cryptonote.org/whitepaper.pdf`, 2013.

[Wui14]   P. Wuille, *BIP62: Dealing with malleability*, Bitcoin Improvement Proposal, 2014, `https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki`.