(Short Paper) A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice

A. Zamyatin[1,2,⋆], N. Stifter[2,⋆], A. Judmayer[2], P. Schindler[2], E. Weippl[2], and W. J. Knottenbelt[1]

[1] Imperial College London, United Kingdom
{a.zamyatin, w.knottenbelt}@imperial.ac.uk
[2] SBA Research, Austria
{nstifter, ajudmayer, pschindler, eweippl}@sba-research.org

Abstract. The loosely defined terms *hard fork* and *soft fork* have established themselves as descriptors of different classes of upgrade mechanisms for the underlying consensus rules of (proof-of-work) blockchains. Recently, a novel approach termed *velvet fork*, which expands upon the concept of a soft fork, was outlined in [22]. Specifically, velvet forks intend to avoid the possibility of disagreement by a change of rules through rendering modifications to the protocol backward compatible *and* inclusive to legacy blocks. We present an overview and definitions of these different upgrade mechanisms and outline their relationships. Hereby, we expose examples where velvet forks or similar constructions are already actively employed in Bitcoin and other cryptocurrencies. Furthermore, we expand upon the concept of velvet forks by proposing possible applications and discuss potentially arising security implications.

1 Introduction

Nakamoto consensus, the underlying agreement protocol of permissionless blockchains, enables eventual consensus on the state updates to a distributed ledger if certain majority assumptions on the hashrate of honest mining participants are upheld [16, 27]. A substantial amount of research has focused on correctly assessing the provided security guarantees, such as the ability for an adversary to succeed in double spending transactions [5, 21, 31]. Despite these remarkable efforts, there still remain open questions and gaps in our understanding of this agreement mechanism. One such topic is approaches for securely changing consensus rules of *permissionless* blockchain protocols [35], such as Bitcoin and Ethereum, which is currently topic of ongoing debate. Reaching agreement on a common set of protocol rules in a decentralized manner could prove to be a problem as difficult as the double-spending problem Bitcoin originally set out to solve.

In this paper we first provide a brief background on core concepts related to this topic after which we discuss and define current protocol upgrade mechanisms considered in permissionless blockchain systems, such as *hard forks* and *soft forks*. In particular, we focus on the recently proposed concept of *velvet forks* by Kiayias et al. [22], which seeks to render protocol upgrades via soft forks more inclusive. We then provide real-world examples where velvet forks or similar concepts are, or have already

⋆ These authors contributed equally to this work.

been, employed. Furthermore, possible negative impacts of such an approach are outlined. In particular with regards to the underlying (game-theoretic) incentive model, such changes may lead to negative side effects in permissionless blockchains. Finally, we suggest the applicability of velvet forks to a number of existing protocol improvement proposals and outline interesting directions for future work.

2 Background

The fundamental mechanism by which Bitcoin and similar *permissionless* blockchain-based systems reach agreement depends, among other, upon consensus participants extending a proof-of-work weighted hash chain, i.e., a *blockchain*. Specifically, it is assumed that a sufficient honest majority of these participants, so called *miners*, will only build upon the branch with the most cumulative proof-of-work, where each element, e.g., *block*, adheres to some pre-agreed set of protocol rules $\mathcal{P}$ under which it is considered valid. The non-deterministic nature of the hash-based proof-of-work employed in such systems, as well as the relatively weak synchrony assumptions of the underlying peer-to-peer network, can lead to situations, where multiple branches are created and extended in parallel. However, the probability of such a blockchain *fork* prevailing for prolonged periods decreases exponentially in its length, if a sufficient majority of miners adhere to protocol rules $\mathcal{P}$ and, in particular, only extend the heaviest chain (known to them) [16, 26]. We use $b \in \mathcal{V}$ to denote a block b is contained in the validity set $\mathcal{V}$ defined by $\mathcal{P}$, i.e., in the set of all blocks considered valid under the protocol rules $\mathcal{P}$.

This brings us to the question how a change $\mathcal{P} \rightarrow \mathcal{P}'$ to the underlying protocol rules may affect this consensus mechanism. *Disagreement* on the validity of a block b under different rules, i.e., $b \in \mathcal{V}$ but $b \notin \mathcal{V}'$, can lead to a *permanent* fork in the blockchain, where a subset of participants will always reject branches building on a, to them invalid, block, regardless of the cumulative proof-of-work these branches accumulate. The requirement for agreement on the block validity also extends to participants not actively involved in the consensus process by mining, such as *fully validating* and *simple payment verification* (SPV) [26] nodes. The former generally adhere to the same full set of rules $\mathcal{P}$ as miners, while the latter only consider a subset $\mathcal{P}_{spv} \subset \mathcal{P}$. For simplification we shall refer to such non-mining participants as *clients*.

3 Mechanisms for Consensus Rule Changes

The term *hard fork* has established itself [8, 18] as a descriptor for protocol changes which can incur a permanent split of the blockchain, as they permit or even enforce the creation of blocks considered invalid under previous protocol rules. As an alternative, *soft forks* intend to retain some level of compatibility to older protocol versions, specifically towards *clients* adhering to previous protocol rules. The concepts of both hard- and soft forks are described in the Bitcoin developer guide [13], as well as the Bitcoin-Wiki [6]. In scientific literature, some of the principal differences between these two types of consensus rule upgrades have been covered in [8, 12, 18]. McCorry et al. furthermore provide a history of forking events in both Bitcoin and Ethereum as part of their work on how parties can bindingly perform atomic cross-chain trades in case of a permanent blockchain fork [25]. A closer description of different protocol forking mechanisms and their relation to each other was also presented in a blog post by Buterin in [9].

Differentiating between Hard and Soft Forks If we consider the possibility of a permanent blockchain split as the defining characteristic of hard forks, most protocol changes would fall into this category. For example reducing the validity set of rules in a protocol update, which is generally considered to be a *soft fork*, can lead to a permanent split in case the majority of consensus participants is not upgraded. Conversely, if an expanding protocol change, i.e. a *hard fork*, does not reach a majority among consensus participants, no permanent fork is actually incurred as upgraded clients will continue to follow the chain with the most cumulative proof-of-work.

This dichotomy helps outline the difficulties in presenting a clear distinction between hard and soft forks. To provide a finer distinction between possible impacts of protocol upgrades and their potential for permanent blockchain forks we present the following classes of protocol changes:

- **Expanding**. The new protocol rules $\mathcal{P}'$ increase the set of blocks V' considered valid with respect to the previous protocol rules $\mathcal{P}$, i.e., $V' \supset V$. Expanding protocol changes can cause a permanent split in the blockchain if the consensus participants adhering to $\mathcal{P}'$ form a majority. However, if a majority retains protocol rules $\mathcal{P}$ no permanent fork occurs as clients adhering to $\mathcal{P}'$ also consider any block under protocol rules $\mathcal{P}$ as valid. Examples include blocksize increase and defining previously unused values as new opcodes.
- **Reducing**. The new protocol rules $\mathcal{P}'$ reduce the set of blocks considered valid with respect to the previous protocol rules. Specifically, the new set of valid blocks V' is a proper subset of the valid blocks of the previous protocol, i.e., $V' \subset V$. Reducing protocol changes represent a soft fork as long as the majority of consensus participants adheres to the new rules $\mathcal{P}'$. If, however, $\mathcal{P}$ retains a majority, a permanent fork is incurred as updated clients and miners will consider some blocks valid under old protocol rules $\mathcal{P}$ as invalid. Examples could be: blocksize decrease, introduction of SegWit (BIP 141 [24]) and removal of an opcode.
- **Conflicting (Bilateral)**. We refer to updates introducing mutual incompatibilities as *conflicting* or *bilateral* protocol changes. Here, the goal is to intentionally cause a permanent fork of the blockchain and prevent potential interactions between the resulting chains, such as the chain ID introduced in Ethereum for replay protection.
- **Conditionally Reducing (Velvet)**. Velvet protocol changes are a special form of update where the new set of *reducing* protocol rules $\mathcal{P}'$ is *conditionally* applied only when the considered elements, such as blocks or transactions, are *valid* under the new rules. Otherwise, the new rules are ignored and previous protocol rules $\mathcal{P}$ are relied upon to determine validity. Since the new rules in $\mathcal{P}'$ are reducing, velvet protocol changes in fact never incur a (permanent) protocol fork as any element considered valid under $\mathcal{P}'$ is also considered valid under $\mathcal{P}$, therefore $V' = V$. Examples, such as P2Pool [3] and overlay protocols, are discussed in Section 4.

Adhering to convention and previous definitions, i.e., [8,9,12], both expanding and bilateral protocol changes are generally considered to be hard forks while reducing protocol changes are referred to as soft forks. In this context the so called *velvet fork* considered in this work would also fall into the latter category of soft forks.

Table 1. Overview of classes of protocol updates $\mathcal{P} \to \mathcal{P}'$. $\mathcal{V}$ and $\mathcal{V}'$ denote the validity sets of old ($\mathcal{P}$) and new ($\mathcal{P}'$) protocol rules, respectively. $\mathcal{N}$ denotes the validity set changes introduced by the protocol update.

Type	Validity Set		Incurred Fork		Examples
	New	Relation to Old	Soft	Permanent / Hard	
Expanding	$\mathcal{V}' = \mathcal{V} \cup \mathcal{N}$, $\exists n \in \mathcal{N} : n \notin \mathcal{V}$	$\mathcal{V}' \supset \mathcal{V}$	never	$\mathcal{V}'$ is majority	Blocksize increase, new opcode
Reducing	$\mathcal{V}' = \mathcal{V} \setminus \mathcal{N}$, $\mathcal{N} \subset \mathcal{V}$	$\mathcal{V}' \subset \mathcal{V}$	$\mathcal{V}'$ is majority	$\mathcal{V}$ is majority	Blocksize decrease, opcode removal, SegWit
Conflicting (Bilateral)	$\mathcal{V}' =$ $(\mathcal{V} \cup \mathcal{N}) \setminus (\mathcal{V} \cap \mathcal{N}) =$ $V \triangle N$	$(\mathcal{V}' \not\subseteq \mathcal{V})$, $(\mathcal{V} \not\subseteq \mathcal{V}')$, $V' \cap V \neq \emptyset$	never	always	Opcode redefinition, chain ID for replay protection
Conditionally Reducing (Velvet)	$\mathcal{V}' = \mathcal{V}$	$\mathcal{V}' = \mathcal{V}$	never	never	P2Pool, merged mining, colored coins

Velvet Forks The velvet fork, as described in [22], does not require support of a majority of participants *and* can potentially avoid rule disagreement forks from happening altogether. In a velvet fork, the new protocol rules $\mathcal{P}'$ are not enforced by upgraded consensus participants and any valid block adhering to the new rules is also a valid block in terms of the old rules. Effectively, velvet forks leverage on the consensus mechanism of protocol $\mathcal{P}$ to bootstrap their own consensus rules $\mathcal{P}'$ which, as part of their rules, produce forward-compatible blocks to $\mathcal{P}$. In principle, protocol updates introduced as velvet forks are always successful, as legacy nodes remain unaware of the changes. However, some protocol updates may not be applicable as a velvet fork, in particular if they require non-upgraded participants to also adhere to the new rules, or the new rules must hold over the span of multiple, possibly arbitrarily many, mined blocks[3].

Other Protocol Update Mechanisms

User Activated Forks. The concept of *user activated soft forks* (UASF) was recently proposed as a mechanism, whereby non-mining participants of the system attempt to take influence on the consensus and protocol upgrade process [4, 9]. We note that user activated forks generally apply to all types of protocol update mechanisms. Specifically, user activated forks aim to incentivize mining participants to perform a consensus protocol upgrade $\mathcal{P} \to \mathcal{P}'$: users and *economic* actors of the system present pledges stating they will strictly enforce the new consensus rules $\mathcal{P}'$ at a certain activation date by their *client software*, regardless of the amount of support of active consensus participants, i.e. miners.

Emergent Consensus. Emergent consensus (EC) is a concept that was proposed as an improvement proposal in the Bitcoin Unlimited client (BUIP001 [34]). Its goal is specifically geared towards reaching dynamic agreement upon the permissible size of Bitcoin blocks, which is currently part of the consensus rules of the Bitcoin protocol. However, the mechanism in principle could also be applied to other protocol rules. EC assumes that a consensus participant will nevertheless accept a, to them invalid,

[3] For example, repurposing anyone-can-spend outputs as is the case with SegWit (BIP 141).

block if sufficient other proof-of-work blocks build upon it. In theory, forks caused by disagreement on the protocol rules could hereby be resolved. However, the resulting impact on security properties is still subject of ongoing discussion and in particular Zhang et al. were able to show models of EC are not incentive compatible, even if all miners fully comply with the protocol [36].

4 Observation of Velvet Forks in Practice

In this section we identify blockchain protocol extensions closely related to velvet forks, which either already have been deployed or whose design follows the same approach.

P2Pool P2Pool [3] is a protocol for implementing decentralized mining pools presented in 2011. In contrast to conventional mining pools, attestation of each miner's contribution to solving the next block's PoW puzzle and the distribution of rewards are accomplished without a trusted operator. P2Pool uses an additional, length-bounded blockchain, the *sharechain*, consisting of otherwise valid blocks which fail to meet the mining difficulty target d but exceed a minimal target d_{share}, agreed upon and determined by the protocol[4], sometimes referred to as *near* or *weak blocks*. These blocks are used to attest each miner's contribution, while the reward distribution is in turn achieved by introducing the following rule: "*Each time a miner finds a block exceeding the target d, she can claim 0.5% of the block reward, while the rest must be distributed among all participating miners according to their portion of the last N sharechain blocks*".

While this additional axiom is an extension to the mined blockchain's rule set, it generates fully backward compatible blocks, and hence remains oblivious to all but P2Pool miners. As a result, any valid block generated by P2Pool miners will be accepted by non-P2Pool miners. In turn, P2Pool miners accept any valid blocks produced by non-P2Pool miners, i.e., even blocks that do not adhere to the above mentioned rule. Since the set of accepted blocks by both parties is exactly the same, P2Pool can be considered a velvet fork.

Sub-chains with Weak Blocks The concept of *sub-chains* was initially proposed by TierNolan (pseudonymous) in 2013 [28] and has been extended, for instance, by Rizun [29]. It builds upon the idea of exchanging *weak blocks* between miners to form sub-chains between consecutive full blocks, by referencing the previous' weak blocks header in an additional pointer.

The required subchain pointer can be readily included in a miner-definable data field, such as the coinbase transaction in the case of Bitcoin. Miners that have adopted sub-chain rules will also accept blocks containing invalid, or no pointer data to sub-blocks. As a result, the set of accepted blocks remains identical for both miners using sub-chains and as those following legacy rules, rendering this proposed protocol extension a form of velvet fork.

Merged Mining Merged mining refers to the process of reusing (partial) PoW solutions from a *parent* blockchain as valid proofs-of-work for one or more *child* blockchains [20]. It was first introduced in Namecoin [7] both as a bootstrapping technique and to mitigate the fragmentation of computational power among competing cryptocur-

[4] The target d_{share} is adjusted such that the sharechain maintains an average block interval of 30 seconds

rencies sharing the same PoW. While a *child* cryptocurrency may require a hard fork to implement merged mining, *parent* blockchains only need to allow for miners to include additional arbitrary data in its blocks. This arbitrary field is then used to link to blocks to the merge mined child cryptocurrency.

Merged mining can be considered closely related to velvet forks, as new consensus rules, namely those of the merge mined children, are incorporated in the parent blockchain in a fully backward compatible way. If either invalid or no links to child blocks are included in a block, the data will be ignored by participants of merged mining and the block is nevertheless accepted. Merged mining and P2Pool make use of the same principle mechanisms, with the marked difference being that in merged mining additional rewards are received in the child cryptocurrency, whereas P2Pool sharechain blocks represent claims to portions of the next valid block's reward on the main chain.

Overlay Protocols and Colored Coins Another concept closely related to the idea of velvet forks is that of overlay protocols and colored coins, inter alia described in [30]. The term colored coin refer to cryptocurrency transactions where the outputs are additionally "colored" to represent some assets or tokens, allowing to use such outputs in transactions to transfer their ownership. We consider colored coins to be part of the class of overlay protocols and herein focus on the latter, more general concept.

Overlay protocols leverage on an underlying property of Bitcoin and similar blockchain systems, namely providing eventual consensus on the ordering of transactions. This primitive, termed *total order broadcast* or *atomic broadcast* has been shown to be equivalent to consensus [11] and can, for instance, be used to readily implement state machine replication. As such, encoding messages in regular valid transactions allows overlay protocols to utilize Bitcoin or similar systems as if they were a (eventual) total order broadcast protocol. While this approach may provide overlay protocols with a mechanism for reaching agreement on the ordering of messages, it does not extend any guarantees towards their *correctness*. In particular, miners may remain completely oblivious to the consensus rules $\mathcal{O}$ of the overlay protocol and only adhere to the underlying base protocol $\mathcal{P}$. Hence, transactions encoding invalid messages of the overlay protocol $\mathcal{O}$, which have to be ignored by the participants of the overlay system, may be included in blocks [8]. However, if participants in the overlay system agree to both the same set of rules $\mathcal{O}$ and the (eventual) ordering of both valid and invalid messages, then ignoring messages considered invalid under $\mathcal{O}$ by all (honest) participants leads to the same (eventually) consistent system state.

Overlay protocols are comparable to velvet forks in that they impose no restrictions and apply new protocol rules $\mathcal{O}$ only if the input is considered valid [5]. The primary difference is that velvet forks additionally assume an active participation in the underlying consensus protocol of $\mathcal{P}$, whereas overlay protocols only take on the role of *clients*. Practical examples of overlay protocols are Omni-Layer (previously Mastercoin) [2] and Counterparty [1].

[5] We point out that the the agreement problem on the overlay protocol rules themselves is hereby of course not solved, and an upgrade $\mathcal{O} \rightarrow \mathcal{O}'$ may cause a logical fork with similar problems to those of the underlying consensus protocol, discussed previously .

5 Considering Security Implications

As outlined in Section 4, velvet forks can be utilized to introduce consensus rule changes in a backward compatible way. However, non-upgraded miners may be unaware of these changes and the potential alterations to the incentives of upgraded *velvet miners* that they entail. As such, blocks produced in accordance with the old rules $\mathcal{P}$ may no longer have the same (economic) utility for velvet miners, as blocks generated under $\mathcal{P}'$, i.e., velvet miners may be biased towards accepting upgraded over legacy blocks. This in turn can have an unclear impact on the security assumptions of such systems, as current attack models mostly do not assume a variable utility of blocks. The following examples outline commonly described attack strategies and how they may relate to velvet forks.

Double Spending The double spending problem was one of the first studied threats in Bitcoin [5, 21, 31]. As miners are required to invest significant amounts of computational power into solving the proof-of-work puzzles, attacks on transactions with sufficient number of confirmations are generally considered economically infeasible. However, long waiting times are often impracticable, while a trade-off between security and usability may be inevitable and must be considered carefully [33]. The necessary thresholds for transaction security assumptions can be shifted in blockchains experiencing a velvet fork, as some blocks may be attributed a higher utility than others by a subset of miners in the system, and must be re-evaluated.

Selfish Mining Selfish mining [15, 32] is known to allow adversaries to increase their expected revenue by deviating from the correct protocol rules. Thereby, selfish miners intentionally withhold blocks and attempt to create a longer secret chain. Determined by the respective strategy, the selfish miner will only publish a select number of blocks from her secret chain, overriding progress in the public chain and forcing honest miners into reorganization. The success rate of the attack is, among other, dependent on the network connectivity of the adversary and the acceptance probability of the blocks in the attacking parallel chain.

However, a velvet fork may significantly impact the success rate of an attacker if some blocks attain a higher probability of acceptance than others, based on the protocol rules they adhere to. In the latter case, an attacker potentially has a higher chance of overriding the public chain, as upgraded miners may prefer her blocks over those of honest (legacy) miners. It is conceivable that the disparity in rewards may even incentivize miners to behave against protocol rules and discard more than one block, i.e., intentionally disregard the heaviest chain rule. Carlsten et al. have shown that selfish mining performs better in Bitcoin under a block reward free model, i.e., when blocks have different economic value and adversaries can utilize this information to better time their attacks [10], and these insights may similarly apply to velvet forks.

Insidious Soft Forks A velvet fork could potentially be abused to enforce a regular soft fork in a hostile manner. Assume a new protocol update $\mathcal{P} \rightarrow \mathcal{P}'$, favored by some portion of the community, does not reach a majority among miners. Hence, it could be deployed as a velvet fork at first. However, if it at some point gains sufficient, i.e, $> 50\%$ adoption, miners adhering to these rules could start to enforce them on the remaining unupgraded participants, i.e., by unilaterally declaring old blocks as invalid and triggering a soft fork. As the velvet miners had sufficient time to accumulate a wider range of support, the fork now has better chances of success due to economic asymmetry, i.e., the unupgraded may conform out of economic interest.

6 Applicability to Existing Proposals

We now move on to provide a non-exhaustive list of consensus extension proposals which could potentially be implemented as velvet forks.

Bitcoin-NG In [14] Eyal et al. present Bitcoin-NG, which aims at improving latency and bandwidth consumption compared to Bitcoin, while maintaining similar security properties. Bitcoin-NG distinguishes between normally mined *key* blocks and so called *microblocks*, which are generated at a significantly lower interval by a *leader*, i.e., the miner of the previous key block. Fees earned from transactions included in microblocks are split in a 40:60 ratio between miners of consecutive key blocks. To disincentivize double spending by malicious leaders, honest leaders can submit *proof of fraud* transactions to the blockchain if they detect attacks, invalidating the funds paid to the adversary.

Velvet Fork applicability Bitcoin-NG adds three rules to the Bitcoin consensus layer, two of which are compatible as a velvet fork. Similar to sub-chains, unupgraded miners would remain agnostic to microblocks, if microblock transactions are included in the subsequent key block[6]. By adding a pointer to the previous Bitcoin-NG key block[7], the new reward scheme can be implemented, despite the presence of legacy blocks. However, the invalidation of funds paid to a malicious leader by a *proof of fraud* transaction must also be accepted by unupgraded miners, which remains an open problem.

Aspen Aspen [17] is an extension to the Bitcoin-NG concept that allows consensus participants to fully validate the correct functionality of blockchain services in a trustless manner, while only keeping track of a, to them, relevant subset of blocks. Service specific data is stored in chains of key and microblocks, thereby, there can exist multiple independent layers of microblock chains used by different services.

Velvet Fork applicability Apart from the rules required to implement Bitcoin-NG, the Aspen protocol requires the annotation of outputs with so called service numbers, which determine on which microblock chains the referenced funds can be spent. In Bitcoin, this could be achievable for instance through using the `OP_RETURN` script opcode. We note that Aspen in its current form could possibly also be implemented using the previously described concept of sub-chains, thereby evading potentially incompatible requirements introduced by Bitcoin-NG.

Extension blocks The Extension block proposal was introduced by Lau [23] and further expanded upon by Jeffrey et al. [19]. It aims at increasing the transaction throughput by introducing an additional layer of (potentially larger) blocks atop the Bitcoin blockchain. While each extension blocks are linked to Bitcoin blocks via the coinbase transaction in a 1-to-1 mapping, they maintain their own independent set of transactions, creating a parallel accounting system.

[6] And do not exceed Bitcoin's block size limitations.
[7] In the data used as input to the proof-of-work of the block.

Velvet Fork applicability To allow users to transfer funds between normal and extension blocks, the proposal re-purposes the `OP_TRUE` opcode, which in turn would render such funds spendable by anyone in the eyes of legacy miners. While the presented approach necessitates a soft fork, alternative constructions possibly allowing for a velvet fork deployment (e.g., multisig locks) may be conceivable.

7 Future Work and Conclusion

Herein, we outlined and extended upon the previously described concept of *velvet forks* and contextualize it to other blockchain consensus rule change mechanisms such as hard- and soft forks. Furthermore we show that variants of this new upgrade mechanism have already been employed in real-world scenarios. Velvet forks present a possible new upgrade path to blockchain consensus rules that could help avoid long-lasting scaling debates and discord in the community. New protocol extensions could be actively deployed without necessitating at least majority agreement by all consensus participants. On the other hand, velvet forks could introduce new possible attacks and threats and fundamentally impact the game-theoretic incentives of the underlying blockchain. In any case, this interesting new concept deserves further research attention.

8 Acknowledgments

This research was funded by Blockchain (GB) Ltd., FFG - Austrian Research Promotion Agency Bridge Early Stage 846573 A2Bit, Bridge 1 858561 SESC, and COMET K1.

References

1. Counterparty. https://counterparty.io/. Accessed: 2017-04-11.
2. Omni layer. http://www.omnilayer.org/. Accessed: 2017-04-11.
3. P2pool. http://p2pool.org/. Accessed: 2017-05-10.
4. Uasf. https://github.com/OPUASF/UASF. Accessed: 2017-04-11.
5. E. Androulaki, S. Capkun, and G. O. Karame. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. In *CCS*, 2012.
6. Bitcoin community. Bitcoin wiki. https://bitcoin.it/. Accessed: 2015-06-30.
7. Bitcoin Wiki. Merged mining specification. https://en.bitcoin.it/wiki/Merged_mining_specification. Accessed: 2017-05-10.
8. J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.
9. V. Buterin. Hard forks, soft forks, defaults and coercion. http://vitalik.ca/general/2017/03/14/forks_and_markets.html, 2017. Accessed: 2017-04-11.
10. M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.
11. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. volume 43, pages 225–267. ACM, 1996.
12. A. Chepurnoy, T. Duong, L. Fan, and H.-S. Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. http://eprint.iacr.org/2017/232, 2017. Accessed: 2017-03-22.
13. B. Community. Bitcoin developer guide - transaction data. https://bitcoin.org/en/developer-guide#transaction-data. Accessed: 2017-04-11.
14. I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Security Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, Mar 2016.

15. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
16. J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.
17. A. E. Gencer, R. van Renesse, and E. G. Sirer. Short paper: Service-oriented sharding for blockchains. *Financial Cryptography and Data Security 2017*, 2017.
18. I. Giechaskiel, C. Cremers, and K. B. Rasmussen. On bitcoin security in the presence of broken cryptographic primitives. In *European Symposium on Research in Computer Security (ESORICS)*, September 2016.
19. C. Jeffrey, J. Poon, F. Indutny, and S. Pair. Extension blocks (draft). https://github.com/tothemoon-org/extension-blocks/blob/master/spec.md, 2017. Accessed: 2017-04-11.
20. A. Judmayer, A. Zamyatin, N. Stifter, A. G. Voyiatzis, and E. Weippl. Merged mining: Curse or cure? In *CBT'17: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology*, Sep 2017.
21. G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Čapkun. Misbehavior in bitcoin: A study of double-spending and accountability. volume 18, page 2. ACM, 2015.
22. A. Kiayias, A. Miller, and D. Zindros. Non-interactive proofs of proof-of-work. Cryptology ePrint Archive, Report 2017/963, 2017. Accessed:2017-10-03.
23. J. Lau. [bitcoin-dev] extension block softfork proposal. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-January/013490.html, 2017. Accessed: 2017-04-11.
24. E. Lombrozo, J. Lau, and P. Wuille. Bip141:segregated witness (consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki, 2012. Accessed: 2017-05-10.
25. P. McCorry, E. Heilman, and A. Miller. Atomically trading with roger: Gambling on the success of a hardfork. In *CBT'17: Proceedings of the International Workshop on Cryptocurrencies and Blockchain Technology*, Sep 2017.
26. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, Dec 2008. Accessed: 2015-07-01.
27. R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 643–673. Springer, 2017.
28. Pseudonymous("TierNolan"). Decoupling transactions and pow. https://bitcointalk.org/index.php?topic=179598.0, 2013. Accessed: 2017-05-10.
29. P. R. Rizun. Subchains: A technique to scale bitcoin and improve the user experience. *Ledger*, 1:38–52, 2016.
30. M. Rosenfeld. Overview of colored coins. https://bitcoil.co.il/BitcoinX.pdf, 2012. Accessed: 2016-03-09.
31. M. Rosenfeld. Analysis of hashrate-based double spending. http://arxiv.org/abs/1402.2009, 2014. Accessed: 2016-03-09.
32. A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.
33. Y. Sompolinsky and A. Zohar. Bitcoin's security model revisited. http://arxiv.org/pdf/1605.09193, 2016. Accessed: 2016-07-04.
34. A. Stone. Bip152:compact block relay. https://github.com/BitcoinUnlimited/BUIP/blob/master/001.mediawiki, 2015. Accessed: 2018-12-01.
35. T. Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf, Apr 2015. Accessed: 2017-10-03.
36. R. Zhang and B. Preneel. On the necessity of a prescribed block validity consensus: Analyzing bitcoin unlimited mining protocol. In *International Conference on emerging Networking EXperiments and Technologies-CoNEXT 2017*. ACM, 2017.