

Research Perspectives and Challenges for Bitcoin and Cryptocurrencies

Joseph Bonneau Andrew Miller Jeremy Clark Arvind Narayanan Joshua A. Kroll Edward W. Felten

Abstract—Bitcoin has emerged as the most successful cryptographic currency in history. Within two years of its quiet launch in 2009, Bitcoin grew to comprise billions of dollars of economic value, even while the body of published research and security analysis justifying the system’s design was negligible. In the ensuing years, a growing literature has identified hidden-but-important properties of the system, discovered attacks, proposed promising alternatives, and singled out difficult future challenges. This interest has been complemented by a large and vibrant community of open-source developers who steward the system, while proposing and deploying numerous modifications and extensions.

We provide the first systematic exposition of the second generation of cryptocurrencies, including Bitcoin and the many alternatives that have been implemented as alternate protocols or “altcoins.” Drawing from a scattered body of knowledge, we put forward three key components of Bitcoin’s design that can be decoupled, enabling a more insightful analysis of Bitcoin’s properties and its proposed modifications and extensions. We contextualize the literature into five central properties capturing blockchain stability. We map the design space for numerous proposed modification, providing comparative analyses for alternative consensus mechanisms, currency allocation mechanisms, computational puzzles, and key management tools. We focus on anonymity issues in Bitcoin and provide an evaluation framework for analyzing a variety of proposals for enhancing unlinkability. Finally we provide new insights on a what we term disintermediation protocols, which absolve the need for trusted intermediaries in an interesting set of applications. We identify three general disintermediation strategies and provide a detailed comparative cost analysis.

I. WHY BITCOIN IS WORTHY OF RESEARCH

Consider two opposing viewpoints on Bitcoin in straw-man form. The first is that “Bitcoin works in practice, but not in theory.” At times devoted members of the Bitcoin community espouse this philosophy and criticize the security research community for failing to discover Bitcoin, not immediately recognizing its novelty, and still today dismissing its importance due to a lack of rigorous theoretical foundation.

A second viewpoint is that Bitcoin hopelessly relies on an unknown combination of socio-economic factors for its current stability which are intractable to model with sufficient precision, failing to yield a convincing argument for the system’s soundness. Given these difficulties, experienced security researchers may avoid Bitcoin as a topic of study, considering it prudent security engineering to only design systems with precise threat models that admit formal security proofs.

We strongly dismiss both of these simplistic approaches and show where each viewpoint fails, forwarding new insights based on multiple examples of existing knowledge. To the first,

we contend that while Bitcoin has worked surprisingly well in practice so far, there is an important role for research to play in identifying precisely *why* this has been possible, moving beyond a blind acceptance of the informal arguments presented with the system’s initial proposal. Furthermore, it is crucial to understand whether Bitcoin will still “work in practice” as practices change. We expect external political and economic factors to evolve, and the system must change if and when transaction volume scales, and the nature of the monetary rewards for Bitcoin miners will change over time as part of the system design. It is not enough to argue that Bitcoin has worked from 2009–2014 and will therefore continue likewise. We do not yet have sufficient understanding to conclude with confidence that Bitcoin will continue to work well in practice, and that is a crucial research challenge that requires insight from computer science theory.

To the second viewpoint, we contend that Bitcoin is filling an important niche by providing a virtual currency system *without any trusted parties* and *without pre-assumed identities among the participants*. Within these constraints, the general problem of consensus in a distributed system is impossible [6], [89] without further assumptions like Bitcoin’s premise that rational (greedy) behavior can be modeled and incentives can be aligned to ensure secure operation of the consensus algorithm. Yet these constraints matter in practice, both philosophically and technically, and Bitcoin’s approach to consensus within this model is deeply surprising and a fundamental contribution. Bitcoin’s core consensus protocol also has profound implications for many other computer security problems beyond currency¹ such as distributed naming, secure timestamping and commitment, generation of public randomness, as well as many financial problems such as self-enforcing (“smart”) contracts, decentralized markets and order books, and distributed autonomous agents. In short, even though Bitcoin is not easy to model, it is worthy of considerable research attention as it may form the basis for practical solutions to exceedingly difficult and important problems.

II. OVERVIEW OF BITCOIN

A. A Contextualized History

We refer the interested reader to existing surveys on the “first wave” of cryptocurrency research [14], [91]. In short, cryptographic currencies date back to Chaum’s proposal for

¹As we shall see, it may not be possible to remove the currency functionality and still have a working consensus system.

“untraceable payments” in 1983 [25], a system involving *bank-issued* cash in the form of blindly signed coins. Unblinded coins are transferred between users and merchants, and redeemable after the bank verifies they have not been previously redeemed. Blind signatures prevent the bank from linking users to coins, providing unlinkability akin to cash.

Throughout the 1990s, many variations and extensions of this scheme were proposed. Significant contributions include: removing the need for the bank to be online at purchase time [26], allowing coins to be divided into smaller units [88] and improving efficiency [24]. Several startup companies including DigiCash [104] and Peppercoin [96] attempted to bring electronic cash protocols into practice but ultimately failed in the market. In fact, no schemes from this “first wave” of cryptocurrency research achieved significant deployment.

Moderately hard “proof-of-work” puzzles were proposed in the early 1990s for combatting email spam [38] (although it was never widely deployed for this purpose [66]). Many other applications followed, including proposals for a fair lottery [47], minting coins for micropayments [97], and preventing various forms of denial-of-service and abuse in anonymous networks [9]. The latter, Hashcash, was an alternative to using digital micropayments (e.g., NetBill [107] and Karma [117]). Proof of work was also used to detect sybil nodes in distributed peer-to-peer consensus protocols [6], and is used in Bitcoin consensus for a similar reason.

Another essential element of Bitcoin is the public ledger, which makes double-spending detectable. In auditable e-cash [102], [103], proposed in the late 1990s, the bank maintains a public database to detect double-spending and ensure the validity of coins, however the notion of publishing the entire set of valid coins was dismissed as impractical (only a Merkle root was published instead). B-money [33], proposed in 1998, appears to be the first system where all transactions are publicly (anonymously) broadcast and stored. Proposed on the Cypherpunks mailing list, b-money received minimal attention from the academic research community.

Smart contracts [111], proposed in the early 1990s, enable parties to formally specify an enforceable agreement using cryptography and scripts. This idea portends Bitcoin’s scripting capabilities.

In 2008, Bitcoin was announced and a white paper penned under the pseudonym Satoshi Nakamoto was posted to the Cypherpunks mailing list [85], followed quickly by the source code of the original reference client. Bitcoin’s *genesis block* was mined on or around January 3, 2009.² The first use of Bitcoin as a currency is thought to be a transaction in May 2010, where one user ordered pizza delivery for another in exchange for 10 000 bitcoins. Since then, increasing number of merchants and services have incorporated Bitcoin in some way, and the price has generally risen, reaching a peak of approximately US\$1200 per bitcoin in late 2013.

Bitcoin’s history has also been colored by association with

²Famously, the first block contains the string “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.”

crime. Bitcoin was famously used in a black market website, Silk Road [27], which operated from Feb. 2011 until Oct. 2013 when it was seized and shut down by the FBI. Botnets have found Bitcoin mining to be a supplemental source of income [52]. A current US federal court case involves a large Bitcoin-based Ponzi scheme [106]. In 2014, a computer virus called CryptoLocker extorted millions of dollars from victims by encrypting their files and demanding a Bitcoin ransom to release the decryption key [44]. Many users’ Bitcoins have been lost due to theft [37] and collapsed exchanges [82].

B. A Technical Overview

We present Bitcoin’s current operation through its three main technical components: transactions (including scripts), the consensus protocol, and the communication network. Bitcoin is exceedingly complex—our goal is to present the system with sufficient technical depth, so the extant literature on Bitcoin, reviewed and evaluated in later sections of this paper, becomes understandable. In particular, a key benefit of our three-component breakdown is that it makes evaluating and systematizing proposed changes (Sections V & VIII) insightful by “decoupling” concepts that may be changed independently.

Sources of information on Bitcoin. Bitcoin can be difficult to define as there is no formal specification. The original Bitcoin white paper [85] provides a good overview of Bitcoin’s design philosophy but many important technical details are omitted or out-dated. The reference implementation `bitcoind` is considered a de facto specification, with further knowledge scattered across a series of “Bitcoin Improvement Proposals” (BIPs), forum postings, online wiki articles, the developer mailing list, and logged IRC discussions.³ We systemize these sources into a precise technical introduction, putting forward the components of the system we consider to be independent design decisions.

1) *Transactions & Scripts*: The state of the world in Bitcoin is represented by a series of messages called *transactions*. Among other possibilities, transactions are foremost published to transfer quantities of currency from one user to another. It is important to note that the large (and growing) list of transactions is the only state in Bitcoin. There is no built-in notion of higher-level concepts such as users, account balances or identities—these all exist only to the extent that they can be imputed by analyzing the list of all published transactions.

Transaction format. A transaction is an array of *inputs* and an array of *outputs*. The entire transaction is hashed using SHA256 and this hash serves as its globally unique transaction ID. Transactions are represented using an ad hoc binary format; this is an early example of an important detail for which `bitcoind` is the de facto specification.

Each output contains an integer value representing a quantity of the Bitcoin currency. The precision of this value

³Which can be found, respectively, at: <https://github.com/bitcoin/bitcoin/bips>, <https://bitcointalk.org/>, <https://bitcoin.it/>, bitcoin-development@lists.sourceforge.net, <irc://freenode.net/#bitcoin-dev>, and <irc://freenode.net/#bitcoin-wizards>

immediately limits the extent to which units of the currency can be sub-divided; the smallest unit is called a *satoshi*. By convention, 10^8 satoshis is considered the primary unit of currency, called one “bitcoin”⁴ and denoted XBT, BTC or ₿.

Each output also has a short code snippet (in a special scripting language) called the *scriptPubKey* representing the conditions under which that transaction output can be redeemed, that is, included as an input in a later transaction.

Transaction scripts. Typically, the *scriptPubKey* specifies the hash of an ECDSA public key and a signature validation routine. This script can be redeemed by signing the entire redeeming transaction using the specified key and is called a “pay-to-pub-key-hash” transaction. The vast majority of Bitcoin transactions are pay-to-pub-key-hash and the system is often described with this being the only possibility, although other transaction types are possible. The scripting language is an ad hoc, non-Turing-complete stack language with fewer than 200 commands called opcodes. They include support for cryptographic operations—*e.g.*, hashing data and verifying signatures. Like the transaction format, the scripting language is only specified by its implementation in `bitcoind`.

Transaction inputs refer to previous transactions by their transaction hash and the index of the output within that transaction’s output array. They must also contain a code snippet which “redeems” that transaction output called the *scriptSig*. To successfully redeem a previous transaction, the concatenated *scriptSig* and *scriptPubKey* must form a program which executes successfully. For pay-to-pub-key-hash transactions, the *scriptSig* is simply a public key and a signature.

Conservation of value. In addition to the requirements that each input of a transaction matches a previous transaction output, and each concatenated script successfully redeems the claimed inputs, transactions are only valid if they satisfy the fundamental constraint that the sum of the values of all transaction outputs is less than or equal to the sum of the values of all inputs. We discuss in Section II-B2 the one exception: the *coinbase* transaction used to create new units of currency.

From transactions to ownership. By themselves, this format of transaction implies several interesting properties. There is no inherent notion of identities or individual accounts which “own” bitcoins. Ownership simply means knowing a private key which is able to make a signature that redeems certain outputs—an individual owns as many bitcoins as they can redeem. Public key hashes, as specified in pay-to-pub-key-hash transactions, effectively function as pseudonymous identities within the system and are referred to as *addresses*. No linking is required to a user’s real-world name or identifying information.

Arguably, there is little that is deeply innovative about Bitcoin’s transaction format. However, the use of a scripting language to specify redemption criteria and the realization that transactions can specify the entire state of the system are non-

obvious design choices given prior cryptocurrency systems, and both have been standard in essentially all subsequent designs. Some proposals extend the semantics of Bitcoin transactions (often by enhancing the scripting language) without changes to any other components.

2) *Consensus and Mining: The need for consensus.* A transaction-based currency system would be insecure if currency were transferred by sending transactions between users. While the signatures would limit only the valid recipient of a previous transaction from referencing it in valid follow-up transactions, there is nothing in the transactions themselves to limit Alice from redeeming some transaction input twice in separate transactions sent to Bob and Carol, both of which would appear valid to Bob and Carol. Bitcoin takes a simple approach to solving this *double spending* attack: all transactions must be published in a global, permanent transaction log and any individual transaction output may only be redeemed in one subsequent transaction. Verifying a transaction now requires verifying the transaction’s scripts as well as ensuring that it is successfully published to the log. In Bitcoin, the log is implemented as a series of *blocks* of transactions, each containing the hash of the previous block, committing this block as its sole antecedent. It is referred to as the *blockchain*.

Note that this design requires global consensus on the contents of the blockchain. If Bob and Carol see two divergent blockchains, they will still be vulnerable to double-spending attacks. One solution is to use a trusted central authority to collect transactions and publish them in signed blocks. However, this is undesirable as this authority might refuse to publish an individual user’s transactions (effectively freezing their assets), might go offline completely, or might intentionally *fork* the blockchain to double-spend coins.

Nakamoto consensus. Bitcoin instead establishes consensus on the blockchain through a decentralized, pseudonymous protocol dubbed *Nakamoto consensus*. This can be considered Bitcoin’s core innovation and perhaps the most crucial ingredient to its success. Any party can attempt to add to the chain by collecting a set of valid pending transactions and forming them into a block. The core ingredient is the use of a challenging computational puzzle (usually given the slight misnomer *proof of work*⁵) to determine which party’s block will be considered the next block in the chain.

The process for choosing a new block is simple: the first announced valid block containing a solution to the computational puzzle is considered correct. Upon hearing it, other participants are meant to turn to finding a followup block. If the found block contains invalid transactions or is otherwise malformed, all other participants are meant to reject this proposed block and continue working until they have found a solution for a valid block. At any given time, the consensus blockchain is the “longest” version. Typically this is simply the branch with the most blocks, but because the mining difficulty

⁴When capitalized “Bitcoin” refers to the entire system whereas lowercase “bitcoin” refers to one unit of currency.

⁵Bitcoin’s mining puzzle is not a true *proof-of-work* scheme but a probabilistic one. Finding a solution is computationally challenging on expectation, but it is possible to get lucky and find a solution with very little work.

can vary between long forks the longest chain must be defined as the one with the greatest expected difficulty to produce.⁶

It is also possible for two valid solutions to be found at approximately the same time (depending on network latency), which leads to a temporary fork during which there are two equal-length chains. Miners can choose either fork in this scenario, and due to the random nature of the computational puzzle, one blockchain will eventually be extended further than the other, at which point the miners will shift to it.

While the original Bitcoin specification provided only an informal argument that eventual consensus would emerge [85], followup work has proved that, assuming an effective and timely broadcast channel and that miners controlling a majority of computational power follow the protocol faithfully, the protocol is robust and the network gradually reaches consensus [43], [80].

Block confirmation. The gradual nature of this consensus mechanism procedure implies that users must wait for blocks to be found in order to gain high confidence that a transaction is permanently included in the blockchain. During a fork, one of the branches will eventually be *orphaned* when miners converge on the other. Typically, both branches will include largely the same set of transactions, but if *conflicting* transactions are included in competing branches then one may be apparently included in the longest branch but then effectively revoked if the other chain branch surpasses it. In the worst case, this will enable the equivalent of a double spending attack [11], [55]. To protect against this risk, users should not consider a transaction to be included until it is in a block which has been “confirmed” by multiple followup blocks.

In theory, users can never be completely sure that a transaction won’t eventually be removed by a very deep fork [12]. However when a majority of miners follow the default protocol, users can infer that a transaction is exponentially increasingly likely (see Section III-A) to end up on the eventual longest chain as more confirming blocks are found. In practice, most Bitcoin clients require 6 confirmation blocks before accepting a transaction as “confirmed.”

Arbitrary-length forks are also prevented in an ad-hoc manner by including hard-coded blockchain prefixes (*checkpoints*) with the default Bitcoin client before which clients will not accept a fork. Laurie [65] argues that the existence of these checkpoints means Bitcoin is not in fact a distributed consensus protocol, and without them eventual consensus would not exist because a future majority miner could always re-write history from the genesis block.

Incentivizing correct behavior. A critical component of this protocol is that a participant who finds a block is allowed to insert a coinbase transaction minting a specified amount of currency and transferring it to an address of their choosing. Because participants to the consensus protocol are working (indeed, racing) to solve this computational puzzle in exchange

for monetary rewards, they are called *miners*. The new currency incentivizes miners to only work towards finding valid blocks, as invalid ones will be rejected by the network and their mining rewards will then not exist in the eventually-longest blockchain. Note that “valid” blocks, from the point of view of miners, are simply blocks which they believe the majority of other miners will accept and build upon, trumping any formal specification of validity (for which there is none beyond the `bitcoind` implementation).

Also note that this consensus algorithm relies on monetary rewards for miners and hence cannot easily be used in systems with no notion of transferable value. In Bitcoin, miners receive all new currency initially and there is no other allowed mechanism for money creation. This is not strictly essential, but the consensus protocol does require some monetary reward is issued to miners or else they have no incentive to find valid blocks and compute the difficult proof-of-work puzzle.

Mining details. The computational puzzle itself requires finding a partial pre-image for SHA-256, a cryptographic hash function. Specifically, the puzzle is to find a block (consisting of a list of transactions, the hash of the previous block, plus an arbitrary nonce value) whose SHA-256 hash is less than a target value. The puzzle is often formulated by the following approximation: finding a hash that starts with d consecutive zero bits.⁷ In so far as the hash output is statistically random, miners can do no better than an exhaustive search over the space of possible nonces for a desired block [9]. The random aspect of this puzzle is important; with a non-randomized proof-of-work function the most powerful individual miner could be expected to find every block first but with a randomized function any miner will have a probability of finding the next block proportional to their share of the competing computational power.

The difficulty of the puzzle is calibrated so that a new block is found, on average, once every 10 minutes. To maintain this, the difficulty is adjusted once every 2016 blocks, or approximately every two weeks, by a deterministic function of the timestamps included in each of the previous 2016 blocks.

Mining rewards and fees. The amount of currency miners may create in each block through a coinbase transaction (the *block reward*) is determined by a fixed schedule. Initially, each block created 50 new bitcoins. This has since halved to $\$25$, and is scheduled to halve roughly every four years until 2140 at which point no new bitcoins will be created.

To enable this wind-down of currency creation, miners do not only profit from block rewards: they are also allowed to claim the net difference in value between all input and all output transactions in this block. For users, a block with greater input value than output value thus includes a *transaction fee* paid to the miners in exchange for publishing their transaction.

To date, transaction fees have primarily been used to discourage overuse of the network and have never provided more than about 1–2% of mining revenue [83]. Fee values have primarily been determined by defaults configured in the

⁶Specifically, this prevents an attacker from forking the blockchain, modifying timestamps on their fork to produce a lower difficulty, and using this lower difficulty to more easily overtake the previous longest chain.

⁷At the time of this writing $d \approx 68$.

reference client [83], with a small number of users opting to pay higher fees in an attempt to have their transactions published more quickly.

Mining pools. In practice, miners often collaborate in mining *pools* [99], although these were not described in the original protocol design and may have been unanticipated. Mining pools are typically administered by a manager who pays miners to mine blocks on their behalf (allocating mining rewards to a key controlled by the pool manager). When blocks are found, the pool manager shares the profits among pool members proportional to the amount of work performed. For example, participating miners can easily prove (probabilistically) the amount of work they have performed by sending “near-blocks” whose hash starts with a large number of zeros (say $d' = 40$) but not enough to make them valid Bitcoin blocks. Pools allow miners to significantly lower the variance in their mining payout, at the cost of a small fee that is paid to the pool manager which lowers their expected total reward. Since 2013, the majority of mining power has been organized into pools. There are several standard protocols for low-latency communication from pool operators to members [90] and between the operators of different pools [30], [68]. While the most popular pools are centrally administered, many miners form *ad hoc* pools using the p2pool protocol [118].

3) *Peer-to-Peer Communication Network*: The final core component of Bitcoin is its communication network. Essentially, it is a decentralized, ad hoc peer-to-peer broadcast network used to propose new transactions and announced newly-mined blocks. Generally, this is the least innovative of the three components and few alternative proposals have made substantial changes.

The performance and stability of the network has an important impact on the consensus protocol for two reasons. First, any latency between the discovery of a block and its receipt by all other nodes increases the possibility of a temporary fork. Fear of frequent forks motivated the choice of 10 minutes as the block creation time in the original design. Second, a malicious miner who is able to control a substantial portion of the network may attempt to favor the broadcast of their own blocks, increasing the likelihood of their blocks “winning” a fork and thus increasing their expected mining rewards. Similarly, any party able to censor the network can selectively block transmissions and freeze assets. Thus it is important for Bitcoin to have a broadcast network which is decentralized (fitting with its overall design), low latency, and where it is difficult to censor or delay messages.

Network topology and discovery. Any node can join the network by connecting to a random sample of other nodes. By default, each node attempts to make 8 outgoing connections, and is prepared to receive up to 125 incoming connections. Nodes behind a NAT, such as mobile clients, are unable to receive incoming connections. Peers who join the network initially need a way to find out about other peers. Like many other peer-to-peer networks, Bitcoin achieves this through the use of dedicated directory servers or “seed nodes,” the identities of whom are hard coded into the reference client;

thereafter, each node maintains a list of peer addresses it knows about. Peers also propagate information about each other through two other mechanisms: when a node establishes a new outgoing connection, it triggers a cascade of relay messages containing its connection information; second, upon receiving an incoming connection, a node asks its peer for a sample from its list of known-about addresses. Overall, the effect of this mechanism is to establish a well-connected random network, with low degree yet low diameter, suitable for rapid broadcast of information through diffusion [35], [56].

Communication protocol. New blocks and pending transactions are broadcast to the entire network by *flooding*. Nodes send INV messages to all of their peers containing the hashes of new blocks or pending transactions whenever they first hear of them. Peers can respond by requesting the full contents of these blocks or transactions if they have not yet seen them (via a GETDATA message). Nodes will: only forward new data once, preventing infinite propagation; only relay transactions and blocks that are valid; only relay the first block they hear of when two blocks are found in a temporary fork; and will not broadcast pending transactions which conflict (double-spend) with pending transactions they have sent. These limits are performance optimizations designed to limit data on the network—a non-compliant node may relay invalid or conflicting data, requiring all nodes to independently validate all data they receive.

Relay policy. By default, Bitcoin nodes only relay transactions and blocks which satisfy *stricter* validation rules than what is permitted by the transaction validity rules. The goal is to prevent various denial of service attacks—an application of the classic robustness principle “be conservative in what you send, be liberal in what you accept.” For example, default nodes only relay transactions containing scripts from a very narrow whitelist of *standard transaction* types. The implication of this policy is that users of the system wishing to have non-standard transactions included in the blockchain cannot use the normal Bitcoin network, but will need to contact an agreeable miner directly.⁸ Another example is that default nodes refuse to relay more than a few thousand transactions below 0.001 XBT per minute as a “penny-flooding” defense.

III. STABILITY OF BITCOIN

A key open question regarding Bitcoin is under what conditions the protocol is *stable*. Stability has been defined in multiple conflicting ways, but it is broadly taken to mean that the system will continue to behave in a way that facilitates a functional currency system as it grows and participants attempt novel attacks. We will consider notions of stability for each component of Bitcoin in turn.

A. Stability of transaction rules

How participants in the Bitcoin ecosystem achieve consensus about the validity rules for Bitcoin transactions is under-analysed. The baseline philosophy is that the rules were set in

⁸For example, Andrychowicz et al. [5] reported needing to submit their complex multiparty lottery scripts directly to the Eligius mining pool.

stone by Satoshi, which we can call *canonicalism*. This has mediated some disagreements about the specified rules, such as a benign bug in the original `OP_CHECKMULTISIG` opcode which has been preserved as canonical.

However, canonicalism cannot fully explain the current rules of Bitcoin. Several changes to the rules have been implemented to add new features (e.g., pay-to-script-hash [2]). Rules have also been modified to fix bugs, with the best-known example occurring in March 2013 when a bug limiting the size of valid blocks was removed. This caused a fork as new, larger blocks were rejected by unpatched clients. To resolve this, the updated clients abandoned a 24-block fork and temporarily ceased including larger blocks during a two-month window for older clients to upgrade [1]. Eventually however, the bug fix won out and unpatched clients, while arguably implementing a more canonical version of the rules, were excluded.

Within the technical rules of Bitcoin, no process is specified for updating or evolving the rules. Without unanimity among miners, any major change may permanently fork the system, with different populations considering the longest blockchain reflecting their interpretation of the rules to be authentic, regardless of its length relative to other blockchains. At this point, it would no longer be clear which version is “Bitcoin.” Thus despite the popular conception of Bitcoin as a fully decentralized system, the need for rule changes (or disambiguation) means some level of governance is inherently required to maintain real-world consensus about what is considered Bitcoin [59].

Currently, de facto governance is provided by the core Bitcoin developers who maintain `bitcoind`, with the Bitcoin Foundation providing a basic organizational structure and raising a small amount of funding through donations to support the development team. As with many early Internet protocols, there is as of yet no formal process for taking decisions beyond rough consensus.

B. Stability of the consensus protocol

Assuming universal agreement on Bitcoin’s transaction rules, various attempts have been made to describe the properties of the consensus protocol which must hold for the blockchain to be stable. We systemize this literature [43], [59], [80], [85] into five desirable stability properties. Note that these have been given different names and different technical definitions by different authors, we will not give formal definitions here but an informal overview of potential stability properties.

- **Eventual consensus.** At any time, all compliant nodes will agree upon a prefix of what will become the eventual valid blockchain. We cannot require that the longest chain at any moment is entirely a prefix of the eventual blockchain, as blocks may be orphaned by temporary forks.
- **Exponential convergence.** The probability of a fork of depth n is $O(2^{-n})$. This gives users high confidence that a simple “ k confirmations” rule will ensure their transactions are permanently included.

- **Correctness.** All blocks in the longest valid proof-of-work chain will only include valid transaction.
- **Liveness.** New blocks will continue to be added containing new valid transactions.
- **Fairness.** Over time, a miner with a proportion α of the total computational power will mine a proportion $\approx \alpha$ of blocks regardless of how they choose transactions to include in their blocks, so long as the blocks are valid according to the rules of Bitcoin.

We say that Bitcoin is *stable* if all of these properties hold. Two analyses have separately argued that Bitcoin is stable assuming a majority of miners executes the default protocol [43], [80]. With billions of dollars now at stake, it is not sufficient to assume that participants will always follow the protocol as specified out of goodwill or inertia.

Surprisingly, correctness is not actually required for a functioning currency, as participants can simply disregard any long proof-of-work chain that contains invalid transactions. However, correctness enables an important performance benefit in the form of SPV clients which validate only proof-of-work and not transactions (see Section VI-A). While they are both often mentioned as desirable properties, neither correctness nor liveness have seen significant analysis or been called into question by plausible attacks; we will not discuss them further.

Incentive compatibility and game theory. The general argument for stability put forward by Nakamoto [85] is that the system will remain stable as long as all miners follow their own economic incentives, a property called *incentive compatibility*. Incentive compatibility has never been formally defined in the context of Bitcoin or cryptocurrencies, but has been used in the economics literature particularly in regards to voting and auction systems. It is an intuitively appealing concept and has proven to have significant marketing value in promoting the stability of the Bitcoin network.

Because we are discussing the interaction of multiple strategic players (the miners), each of whom is presumed to be optimizing an objective reward function, formal analysis could be attempted using the framework of game theory [86], [119]. We assume each miner chooses a *strategy*, which is a (possibly randomized) function from states of the game to game moves. Miners’ strategies will lead to a distribution over resulting blockchains, and each player’s payoff will depend on the resulting blockchain.⁹

We call the strategy of following the default mining rules (see Section II-B2) the *compliant* strategy. This is sometimes called “honest” but we avoid this term as non-compliant strategies might also reasonably be considered honest.

Recall that a *Nash equilibrium* is the state of a strategic game in which all players, knowing the actions of all other players, believe their choice of strategy to be the best response to the strategies of the other players. That is, at a Nash equilibrium, no player benefits by changing to a different strategy. If we could prove that all miners playing the compliant strategy

⁹Even if all miners’ strategies are deterministic, the Bitcoin mining process is inherently randomized, so the outcome will always be a distribution.

is a Nash equilibrium, this would imply incentive compatibility for Bitcoin (and hence stability), as no miner would have any incentive to change strategy.

We only call this *weak stability* though, because in general games may have arbitrarily many Nash equilibria. It is possible that universal compliance is one equilibrium, but the system could eventually degenerate into a different equilibrium which does not result in stability for Bitcoin. If we could prove that universal compliance is the only Nash equilibrium, this would imply *strong stability* in that the system would eventually work its way back to this state.

1) *Results on stability:* We discuss what is known on Bitcoin stability under various assumptions. All models in this section assume that miners' objective function is purely obtaining nominal bitcoins, but we will return to that assumption.

Supermajority compliance implies stability. The original Bitcoin white paper considers an attack in which a malicious miner tries to reverse a transaction by "trying to generate an alternate chain faster than the honest chain." The white paper claims that such an attack will fail by modeling the race between the attacker and the rest of the network as a binomial random walk.

Miller and LaViola [80] and Garay et al. [43] provide independent formal proofs that if a supermajority of miners follow the compliant strategy and communication latency is negligible compared to the expected time to discover a block, miners will eventually agree on an ever-growing prefix of the transaction history regardless of the strategy of non-compliant miners. This is sufficient to ensure stability, with the size of this supermajority required varying slightly depending on network and other assumptions.

Simple majority compliance does not guarantee fairness. Eyal and Sirer [41] analyzed a strategy they refer to as *selfish mining*¹⁰ in which a miner temporarily holds blocks after finding them, hoping to find itself two blocks ahead of the longest publicly-known chain so that it can effectively mine unopposed until the remainder of the network has caught up to within one block at which point the selfish miner will publish their withheld blocks. If all other miners are compliant, this strategy is always advantageous for a miner controlling at least $\frac{1}{3}$ of all mining power. This clearly violates the fairness property as the selfish miner would earn an outsize share of mining rewards at the expense of others. While convergence would be slightly slower, it appears rapid convergence would still apply as the chance of a very long fork due to withheld blocks is still exponentially small.

This strategy may also be advantageous for attackers with even lower levels of mining power, depending upon assumptions about network propagation and how miners will choose between near-simultaneously announced blocks. Garay et al.'s model [43] naturally incorporate selfish mining and worst-case assumptions about an attacker's ability to win simultaneous-block broadcasts and shows that under these assumptions selfish mining is always profitable, meaning fairness inherently

relies on assumptions about the communication network. These results imply that universal compliance is not a Nash Equilibrium for many distributions of mining power, including some which have been observed in practice, but there is no evidence that selfish mining has ever been attempted.

Weak stability is guaranteed under strong assumptions. Kroll et al. [59] analyzed a model in which there is no majority, no collusion, and miners are assumed to have perfect information about all discovered blocks (negating block withholding). In this model, universal compliance is a Nash Equilibrium (although not unique), implying that Bitcoin is (weakly) stable.

With a majority miner, stability is not guaranteed. It is well known that if a single miner controls a majority of computational power, stability is not guaranteed. The majority miner could collect *all* of the mining rewards, simply by ignoring blocks found by others and building their own chain which by assumption will grow longer than any chain the rest of the miners can make. This would only necessarily undermine fairness, but the majority miner could undermine the other stability properties as well if they desired. For example the majority miner might also intentionally introduce arbitrarily long forks in the block chain to reverse (and hence double-spend) transactions for profit, violating the eventual consensus and convergence properties.

If miners can collude, stability is not known. Even in the absence of a majority miner, multiple smaller miners could potentially collude to form a cartel controlling a majority of mining power and emulating any strategy available to a single majority miner. It is not known whether such a cartel would be stable or whether it would fail because cartel members would cheat each other or excluded miners could break it up by offering to form a cartel on more favorable terms. If a stable cartel did form, similar to the majority mining scenario blockchain stability could not be guaranteed without strong assumptions about the cartel's objective function. Mining pools could possibly be a technical mechanism for cartel formation; the dynamics of miners' choice of pools and migration between pools have not been studied. It also appears no rigorous analysis has been attempted of whether and how miners might encourage others to participate in a majority attack cartel through side-payments.

Future directions and attacker rationality. These results do not provide convincing justification of Bitcoin's observed stability in practice nor convincing assurance of its continued stability in the future. An underlying problem is that miners are clearly not solely interested in obtaining nominal bitcoins but in obtaining real-world profits. Some non-compliant strategies, particularly those that would affect stability in a visible way, might undermine public confidence and weaken demand for bitcoins. Indeed, in practice the exchange rate has been found to dip in the face of technical glitches with the system [67].

Thus, although one strategy may earn more nominal bitcoins than another, if it drives down the exchange rate it may provide a lower effective reward for miners. We call this principle *exchange-rate rationality*.

In practice most miners have a significant additional interest

¹⁰A related set of strategies were exhibited concurrently by Bahack [10].

in maintaining Bitcoin's exchange rate because they have significant capital tied up in mining hardware which will become less if the exchange rate declines. If miners expect they will maintain their share of mining power far into the future with low marginal costs (*e.g.*, if a substantial portion of their operational costs are paid upfront to buy equipment), then they may avoid strategies which earn them more bitcoins but decrease the expected value of their future mining rewards. We call this principle *long-term rationality*.

Indeed, Nakamoto dismissed the possibility of majority-miner attacks in the original Bitcoin paper [85] by an appeal to long-term rationality, arguing that they would permanently damage the system (and exchange rate) and "playing by the rules" would be more profitable over time. In practice, the GHash.IO mining pool exceeded 50% of the network's computational capacity for an extended period in July 2014 and publicly promised to not attack and to shrink in order to avoid damaging confidence in the system.

It may be that one or both rationality assumptions are required to model the lack of attacks on Bitcoin. Unfortunately, this is very difficult to capture in any tractable game-theoretic model as reasoning about the exchange rate inherently depends on human judgement and market confidence. Modeling this assumption more formally is a significant open problem.

Another lingering model problem is that all of these models have assumed each block carries a constant, fixed reward fee, although the planned transition to transaction fees will negate this assumption and require more complex models which take into account the distribution of available transaction fees.

2) *Results with non-monetary objective functions*: At least two strategies have been analysed which may be advantageous for a miner whose objective are not purely monetary.

Goldfinger attacks. If a majority miner's goal is explicitly to destroy Bitcoin's stability and hence its utility as a currency, they can certainly do so by intentionally introducing deep forks, undermining convergence. Kroll et al. [59] introduced this model and named it a *Goldfinger attack*. For example, a state wishing to damage Bitcoin to avoid competition with its own currency, or an individual heavily invested in a competing currency, may be motivated to attempt such an attack. Arguably, these attacks have already been observed through *altcoin infanticide*, in which deep-forking attacks against new competing currencies with low mining capacity have been successfully mounted by Bitcoin miners.¹¹

Feather-forking. Miller [77] proposed the strategy of *feather-forking*, in which a miner tries to censor a set of target transactions by publicly promising that if a targeted transaction is included in the block chain, the attacker will retaliate by attempting to fork the block chain, ignoring the block containing the targeted transaction. The attacker's fork will continue until it either outraces the main branch and wins, or falls behind by k blocks causing the attacker to rejoin the main and accept publication of the targeted transaction. An

attacker with $\alpha < 50\%$ of the mining power will succeed with probability $\frac{\alpha^2}{1-\alpha+\alpha^2}$ and will, on expectation, lose money.

However, if the attacker can convincingly show that they are serious about performing the retaliatory forking, other miners will gain by shunning the targeted transactions as they also lose on expectation if the attacker does retaliate. Thus, an attacker may be able to enforce their blacklist with no cost at all, as long as all other miners are convinced the attacker will perform a costly feather-forking retaliation. This has some resemblance to the well-studied Chicken game from game theory [95], in which players have an incentive to convince the other they will take a self-destructing action if the other does not yield. Feather-forking has never been observed in practice, which may again be attributed to long-term rationality.

3) *Stability of mining pools*: Mining pools rely on participants to submit valid blocks when they are found and are vulnerable to participants submitting partial shares in exchange for compensation but withholding valid blocks to lower the pool's profitability. Though this attack has long been known, it appears self-destructive as the participant withholding a block is lowering their own earnings in addition to other pool members. However, it has been shown [31] that a large miner (or a pool) can actually profit from using some of its mining power to *infiltrate* another pool by submitting partial shares but withholding valid blocks. The benefit is that the capacity used to infiltrate will not contribute to increasing the difficulty of the mining puzzle (as blocks are not published) but can still earn profits. This strategy is advantageous to a large miner or pool across a range of mining capacities for the attacker and the infiltrated pool.

Eyal [40] provides an extended treatment of this attack and shows that it, between any two pools, the resulting game is an iterated prisoner's dilemma, with a Nash equilibrium of both pools attacking each other but a Pareto equilibrium of neither attacking. This attack can be detected statistically if done on a large scale, which has happened at least once in the wild against the Eligius pool in June 2014 [121]. However, a clever attacker can easily obfuscate the attack using many participant addresses. Further countermeasures have been proposed but not seriously studied or deployed. As an iterated prisoner's dilemma, it is possible pools will avoid attacking each other through out-of-channel communication and the threat of retaliation.

C. Stability of the peer-to-peer layer

Almost all analysis of Bitcoin assumes that the peer-to-peer layer functions as specified and that, in general, a majority of participants will learn nearly all of the available protocol state information within reasonable time scales. However, Babaioff et al. [7] demonstrated that information propagation at the peer-to-peer layer is not always incentive compatible for protocol participants. It remains open whether participants internalize sufficient value from the peer-to-peer network as a public good to justify the opportunity costs of propagating information Babaioff et al. identified, or whether the information propagation equilibrium observed in the wild (in which people

¹¹For example, CoiledCoin was an altcoin that was destroyed by a significant attack from Eligius, a Bitcoin mining pool [71].

willingly participate in the peer-to-peer protocol) is unstable and might break down eventually.

Johnson et al. [54], [63] study whether and when participants in the peer-to-peer protocol are incentivized to engage in network-level denial-of-service attacks against other players. Johnson et al. conclude that mining pools have an incentive to engage in attacks, that pools have a greater incentive to attack larger pools than smaller pools and that larger pools have a greater incentive than smaller pools to attack at all. Denial-of-service attacks against pools are regularly observed in the wild, so this theoretical analysis can be backed up by observed phenomenology [116]. Others have performed measurement and simulation studies to determine the dynamics and time scale of information propagation [35], [36].

IV. MODIFYING BITCOIN

The remainder of this paper will largely form several comparative evaluations of proposed changes and extensions to Bitcoin. The following levels of changes are distinguished:

- **Hard forks.** A protocol change requires a hard fork if it enables transactions or blocks which would be considered invalid under the previous rules, such as increasing the miner block reward, changing the fixed block size limit, or removing an opcode. If miners update to the new protocol, they may produce blocks that are rejected by other nodes leading to a permanent (or “hard”) fork. Changes involving a hard fork therefore require near-unanimity to be attempted in practice.
- **Soft forks.** In contrast to a hard fork, a soft-fork change is one that’s backward compatible with existing clients; generally this involves a restriction of which blocks or transactions are considered valid. Such a change requires only the support of a majority of miners to upgrade, since older clients will continue to consider their blocks valid. A miner that doesn’t upgrade may waste computational work by generating blocks that the rest of the network considers invalid and ignores, but will always rejoin the longest chain found by the majority of the miners. In some cases, a soft fork can be used to introduce new opcodes to the scripting language. This is possible because there are currently several *unused* opcodes that are interpreted as no-ops; including these in a transaction output may make it spendable by *anyone*, and hence they are typically avoided. However, any one of these op-codes can be given new semantics if miners decide to reject transactions that fail some condition indicated by this opcode. This is a strict narrowing of the set of acceptable transactions, and hence requires only a soft fork.
- **Relay policy updates.** Recall from Section II-B3 that nodes enforce a stricter policy in what they will relay than what they will actually accept as valid. Changing this policy or most other aspects of the communication network require the least coordination as they can typically be done in a backwards-compatible fashion with nodes advertising their protocol version number. The default relay policy has al-

ready changed several times to add new standard transaction types such as multi-signature transactions.

A. Altcoins

Due to the friction of changing Bitcoin, hundreds of derivative systems, referred to as *altcoins*, have arisen to incorporate alternate design approaches. Many of these systems have forked Bitcoin’s codebase and maintained most of its features, although some systems (such as Ripple) are completely independent designs. Altcoins must bootstrap the initial allocation of currency to entice users to participate, which can be achieved in several ways:

- **New genesis block.** Altcoins may simply start a new blockchain from scratch, allocating funds to initial miners as Bitcoin did in its early days. This approach is now viewed suspiciously by the cryptocurrency community due to a wave of altcoins allegedly launched by founders hoping to cash in through early mining.
- **Forking Bitcoin.** To avoid this suspicion, an altcoin might intentionally choose to fork Bitcoin at a certain point, accepting the prior transaction history and ownership of funds. Bitcoin owners would continue to have bitcoins in the original system, plus an equal amount of the new currency at the time of its founding. This would function like a hard fork, but with no claim that the new system is the legitimate Bitcoin system. Interestingly, this approach seems not to have been attempted seriously.
- **Proof-of-burn.** A more popular approach to inheriting Bitcoin’s allocation of resources is *proof-of-burn* [110], in which users must provably destroy a quantity of Bitcoins—*e.g.*, transferring funds in Bitcoin to a special address whose private key cannot be found, such as the key with a hash of all zeroes. This approach has the downside of permanently lowering the number of units of Bitcoin available for use.
- **Pegged sidechains.** Most recently, a number of influential Bitcoin developers [8] proposed *sidechains*, to which bitcoins can be transferred and eventually redeemed. Adding validation rules to redeem currency from a sidechain would require at least a Bitcoin soft-fork, and hence is not yet possible.

Altcoins also must compete with Bitcoin for miners (and avoid Goldfinger attacks by Bitcoin miners), which can be difficult prior to the currency achieving a non-zero exchange rate. A popular approach is *merge-mining*, whereby an altcoin accepts blocks if their root is included in a valid Bitcoin block, thus enabling Bitcoin miners to mine blocks in the altcoin without performing any additional work. This can quickly provide an altcoin the full mining power of Bitcoin, as many Bitcoin miners now merge mine a large number of altcoins to earn extra rewards. However, it limits the ability of the altcoin to deviate from Bitcoin’s computational puzzle.

V. ALTERNATIVE CONSENSUS PROTOCOLS

Bitcoin’s consensus protocol has been its most heavily debated component, due to the open questions about stability (see Section III-B), concerns about the performance and scalability

of the protocol [109], and concerns that it is wasteful of computation. In this section we evaluate alternative proposals for consensus, noting that in each case the stability implications of the proposed changes are unknown and alternative proposals rarely define any specific stability properties they claim to provide.

Typically, alternate consensus schemes aim to fix some specific perceived problem with Bitcoin and hope the stability arguments towards Bitcoin will carry over to the new consensus protocol, although given the lack of a solid model guaranteeing stability for Bitcoin this may be a shaky assumption.

A. Parameter changes

Bitcoin’s consensus protocol incorporates many “magic constants” which were hard-coded based on initial guesswork. Nearly every altcoin has varied at least some of these parameters, yet the modifications are often controversial and we still have only a few clear guidelines on how these should be set and how sensitive the consensus protocol is to their change.

Inter-block time and difficulty adjustment window. Bitcoin automatically adjusts the difficulty of its proof-of-work puzzle so that each puzzle solution is found (on average) ten minutes apart. This setting is constrained primarily by network latency, which plays a critical role. If this rate is too high, then miners will frequently find redundant blocks before they can be propagated. On the other hand, a slower block rate directly increases the amount of time users need to wait for transaction confirmations. Bitcoin’s setting is by all accounts *conservative*; all altcoins we know of have the same or *faster* blocks (e.g., the second most popular system, Litecoin, is four times faster). There are many proposals to modify aspects of the communication network to reduce latency, allowing this parameter to be safely reduced [35], [109].

Limits on block and transaction size. One of the most controversial proposed changes is to change the fixed 1-megabyte limit on the size of a block [3]. As the transaction volume has steadily increased, this limit may soon be regularly reached. The upper bound on transaction volume is currently only 7 per second, approximately 1,000 times smaller than the peak capacity of the Visa network [49]. Once this limit is reached, transactions will effectively need to use their fees to bid for a scarce resource. This may raise the cost of using Bitcoin, potentially slowing adoption, yet increasing the revenue for miners. It may also lead users to rely on intermediaries who aggregate and settle transactions off-chain. The limit is artificial, and the network’s bandwidth could likely sustain an increase; on the other hand, increased transaction volume may exclude some participants who are bandwidth-limited. Several altcoins have raised this limit in their specification, though to our knowledge none has come close to actually utilizing this capacity so it remains unknown how it will affect operation of the system.

Monetary Policy. Bitcoin’s consensus protocol effectively mandates a monetary policy in the rate at which new currency is minted and the schedule by which this rate changes. By mandating a capped amount of currency, Bitcoin effectively

has a *deflationary* monetary policy which has caused multiple economists to predict the system will eventually be destabilized by a *deflationary spiral* in which nobody is willing to spend bitcoins as hoarding them is considered more profitable [48], [60]. Issuance of coins is one of the most widely varied parameters: for example, in Dogecoin inflation will continue indefinitely but at a harmonically-diminishing rate while in Freicoin [42], the inflation rate maintains constant.

B. Alternative computational puzzles

Miller et al. [81] present a formalism for Bitcoin-compatible proof-of-work schemes called scratch-off puzzles, which essentially must be decomposable into individual attempts. This property is often referred to as “progress-free;” it guarantees that a) the creator of each block is chosen by an approximately weighted random sample of computational power, b) even individual participants are able to receive (proportional) reward for their contribution, c) the time between consecutive puzzle solutions is sufficiently large that puzzle solutions propagate. Bitcoin’s SHA-256 puzzle satisfies these properties, and many other constructions are possible.

ASIC-resistant puzzles. While Bitcoin mining was originally performed using ordinary general-purpose processors, the competitive nature of mining has led to a steady movement towards more powerful and energy-efficient customized hardware. Today, most of the computational power is accounted for by ASICs. Taylor provides an excellent survey of the technical challenges in computing SHA-256 efficiently at scale and estimates that today’s ASICs are already within an order of magnitude of theoretical efficiency limits [112].

This has both positive and negative implications: a positive is that “botnets” who steal cycles from commodity equipment are no longer competitive against modern mining rigs [52]; a (perceived) negative is that this moves Bitcoin mining away from its core democratic value (i.e., “one-CPU-one-vote” [85]) since most participants in the system do not own ASICs and hence perform no mining at all.

In response, many proposals have been made for ASIC-resistant mining puzzles. Ideally, an ASIC-resistant puzzle could be effectively solved using commodity hardware, with only minor performance gains for customized hardware. The primary approach taken so far has been to design “memory-hard” puzzles which are designed to require efficient access to a large memory. The most popular memory-hard puzzle so far (used in Litecoin and Dogecoin, among others) has been the scrypt hash function [92] originally designed for cracking-resistant password hashing. Until 2014 it was unknown if it is possible to design a puzzle which is memory-hard to compute but memory-easy to verify. Tromp’s cuckoo-cycle puzzle [114] appears to answer this question affirmatively.

It remains an important open problem if ASIC-resistance is possible.¹² ASICs that mine scrypt, for example, have

¹²This problem has applications in other applications including password hashing and password-based encryption, towards which the current Password Hashing Competition is attempting to identify a new standard.

already been released in the market and offer performance improvements comparable to SHA-256 ASICs.

Useful puzzles. Achieving consensus through computational puzzles appears to be wasteful both in terms of the energy consumed in computation and the energy and resources used to manufacture mining equipment. Becker et al. [13] posit that ultimately Bitcoin might be subject to control by real-world entities in control of the world’s energy supplies. If it is possible to obtain the same level of security while utilizing the work for some additional purpose, then some of this cost can be recovered.

A common suggestion is to use a search function with applications to scientific research, such as the popular Folding@Home [62] project. A challenge for useful puzzles is that they must be automatically generated and verified with no trusted parties, (otherwise this party could choose puzzles on which they already had a head start). Kroll et al. [59] further argued that any useful puzzle must produce a pure public good, or else it might increase the amount mining by the amount it recovers, canceling out any recycling effect.

Primecoin [57] introduced the first useful puzzle in a successful altcoin. Its puzzle requires finding sequences of large prime numbers of mathematical interest and which may be used as parameters for cryptographic protocols. Miller et al. [79] proposed a puzzle incorporating proof-of-retrievability, so that mining requires storing a portion of a large public dataset. In particular, if the public dataset is of use to the Bitcoin network itself (e.g., the blockchain history), this approach provides additional incentives to contribute resources to the network.

Nonoutsourcable puzzles. The growth of large mining pools [72] and their potential to facilitate collusion and cartel formation has motivated the design of puzzles which cannot be easily outsourced. Members of a pool do not inherently trust each other; instead, these coalitions succeed because members can easily prove that they are performing mining work that, if successful, would pay the reward to the pool manager. Miller et al. [81] as well as Siler and Eyal [108] have proposed “nonoutsourcable” variations of scratch-off puzzles that ensure whoever performs the mining work can claim the reward for themselves when a block is found, thus thwarting pools’ enforcement mechanisms and making the formation of large pools between anonymous participants unlikely.

C. Virtual Mining and Proof-of-Stake

At a high level, proof-of-work schemes exist to require expenditure of resources to perform mining. Instead of expending external computing resources, it may be possible to expend *wealth* directly. Instead of having participants “mine” by exchanging their wealth for computational resources, they may simply exchange their wealth for the ability to choose blocks. Rather than advancing the global history by a random sample of participants weighted by computational power, the random sample is weighted by the previous allocation of wealth. We can call such schemes broadly “virtual mining.”

The earliest virtual mining proposal [94] used the term “proof-of-stake.”

Virtual mining offers two main benefits: first, it may be more difficult for an attacker to acquire a sufficiently large amount of the digital currency than to acquire sufficiently powerful computing equipment. Second, by avoiding the consumption of *real* resources (i.e., compute cycles), no real-world resources are wasted. There have been several variations of virtual mining proposed to date, which vary mainly on the criteria by which possession of a quantity of currency makes one eligible to choose the next block:

- **Proof-of-coin-age.** Peercoin [58] proposed mining by demonstrating possession of a quantity of currency by posting a transaction (potentially to oneself, in which case the coins are not lost). Each quantity of currency is weighted by its “coin-age”, the time since the coins were last moved.
- **Proof-of-deposit.** In Tendermint [61], participation in mining requires depositing coins in a time-locked bond account, during which they cannot be moved.
- **Proof-of-burn.** Stewart [110] proposed mining by *destroying* coins (i.e., sending them to an unspendable address).
- **Proof-of-activity.** Bentov et al [19] proposed having every coin owner implicitly entered into a mining lottery by default; periodically, random values from a beacon (e.g., generated from transactions occurring on the network) are used to select randomly among all the coins in the system; the current owner of the winning coin must respond with a signed message within some time interval.

There has yet to be any formalization of the model assumptions that may allow virtual mining systems to achieve security, or to compare virtual mining systems to proof-of-work systems in a common setting. Poelstra [93] presents a survey of the folklore arguments suggesting that consuming external resources (i.e., burning energy) is necessary for blockchain security and hence virtual mining schemes are inherently infeasible. The central argument – deemed the *nothing-at-stake* problem – is that virtual mining is susceptible to costless simulation attacks; it costs nothing to construct an alternate view of history in which the allocation of currency evolves differently. Providing a rigorous argument for or against stability of virtual mining remains an open problem.

D. Designated Authorities.

Although Bitcoin’s decentralized nature is one of the primary selling points and is a fiercely-defended principle among many in the community, we can observe that Bitcoin’s consensus protocol could be drastically simplified if we could rely on a (small) number of designated authorities to receive, sequentially order, and sign transactions. This would make stability assumptions much easier to reason about and remove concerns about wasteful computation all at once. We might even invoke a similar long-term rationality argument to that used for Bitcoin and argue that if the authorities earned a small income by behaving honestly they would have no incentive to misbehave.

Trust in these authorities might be limited by using a mutually untrusting set of authorities [64], using social networks to choose which authorities to trust [105] or provisioning for coin owners to choose their trusted authorities every time they spend coins [22]. Ripple [105] is one of the few altcoins launched with this model; however, its stability argument remains essentially unproven.

VI. CLIENTSIDE SECURITY

A. Simplified Payment Verification (SPV) Security

Although the reference Bitcoin client maintains a validated copy of the entire blockchain, this would impose a prohibitive burden on mobile devices. A simple observation leads to a lightweight alternative: assuming that a majority of nodes only mine on valid chains (the *correctness* property of Section III-B), then clients need validate only the proofs of work and can trust that the longest chain only contains valid transactions. Such SPV proofs [85] enable untrusted nodes to efficiently prove to lightweight clients that a transaction has been included in the agreed-upon history.

SPV is implemented in the BitcoinJ library which underlies most mobile (and some desktop) Bitcoin clients. Though effective in practice, this technique still requires lightweight clients to maintain an ever-growing chain of proof-of-work solutions.¹³ SPV also requires disclosing the set of addresses the client is interested in to untrusted nodes, a practical privacy concern (see Section VII and [45]).

B. Key Management

Bitcoin assumes a solution to the longstanding problem of usable public key cryptography for user authentication, while nearly all other forms of online commerce today rely on passwords or confidential credit card information. Developers of Bitcoin software have attempted a variety of approaches solve, or at least mask, the complexities of key management. Eskandari et al. [39] propose a set of evaluation criteria for the usability Bitcoin key management interfaces and conclude that current tools employ complex metaphors which don't fully capture the implications of key management actions.

Keys stored on device. Keys stored on a device in a file is a simple model, but may be stolen by specifically-crafted malware [69], may be inadvertently shared (network sharing, offsite backup, P2P filesharing), or the device may be physically stolen or lost. Some clients send change to newly created Bitcoin addresses, requiring a new backup each time the keypool is depleted (generally without any user-interface indication when it happens), while others send change to the originating address or derive all keypool addresses from a random seed.

Password-protected wallet. A Bitcoin client may allow a stored keypool file (called a wallet) to be encrypted with a key derived from a user-chosen password. Password-protected wallets deter certain types of theft, additionally requiring password guessing or keystroke capture if the file is physically or

digitally stolen. Password-protected wallets may also mislead the user to believe that the password itself provides access to their funds (*e.g.*, on a new device).

Offline storage. To further enhance theft-protection from malware-based threats, wallets can be stored offline on some form of passive portable media, such as paper or a USB thumbdrive. This enables the use of traditional physical security to protect the media, which users may have a better mental model of. In the case of paper, typically private keys are printed in scannable form (*e.g.*, QR codes). Unlike paper money, funds can be stolen by passive observation (*e.g.*, on live television [98]). Offline storage must be updated each time the keypool is depleted. Finally, as offline wallets are not immediately accessible for use, they must be loaded into a device at some point and then again become susceptible.

Air-gapped and hardware storage. Air-gapped storage is a special case of offline storage, where the device holding the keypool can perform computations, such as signing transactions for the keys it holds. Air-gapped devices can thwart certain types of thefts by never exposing the keypool directly to an internet-connected device. That said, unauthorized access to a transaction-signing oracle is not much different from accessing the keypool itself—both allow theft. Related, hardware security modules (HSMs) emulate the properties of an air gap by isolating the key material from the host device, and only exposing the ability to sign transactions.

Password-derived wallet. A Bitcoin client may permit deterministically deriving a keypool from a user-chosen password. This allows the keypool to be regenerated from a memorized password on any new device. The primary drawback of a password-derived wallet is that weak passwords can be found through unthrottled exhaustive search, as a fingerprint of the associated public key will be in the ledger if the account holds any amount of Bitcoin. Additionally, a forgotten password will orphan all funds in the account.

Hosted wallet. A final approach is storing your keypool with a third party webservice that provides transactional functionalities through standard web authentication mechanisms, such as a password or two-factor authentication, and allow a password recovery mechanism. This provides the closest experience to traditional online banking, however requires trusting the host: the host may lose or steal from the accounts it hosts (many incidents are catalogued online [37]) and include over 40 events involving losses greater than 1000 XBT).

VII. ANONYMITY & PRIVACY

Bitcoin provides a limited form of unlinkability: users may trivially create new pseudonyms (addresses) at any time. This was argued in the original specification to provide strong privacy [85], however it quickly became clear that due to the public nature of the blockchain it sometimes possible to trace the flow of money between pseudonyms and conclude that they are likely controlled by the same individual.

A. Deanonimization

The actual level of unlinkability depends heavily on use patterns and implementation details that we term *idioms of use*,

¹³In theory, succinct proof systems (*e.g.*, SNARKs [16]) could reduce this verification cost to a constant.

following [75]. For example, merchants that generate a fresh payment address for each sale ensure that received payments are not automatically linkable on the blockchain. By contrast, the customer may need to assemble the payment amount from multiple addresses she owns,¹⁴ linking these addresses (and their accompanying transactional history) together on the blockchain, given different users rarely contribute inputs to a single, joint transaction.¹⁵ Other idioms such as “every non-change output is controlled by a single entity” [4] and “an address is used at most once as change” [75] can also be utilized by an adversary to link together different addresses controlled by the same entity.

Linking can be applied transitively to yield clusters of addresses; this is an instance of *transaction graph analysis*. A major challenge for the adversary is that these idioms are fragile—they may yield false positives and may lose accuracy over time as implementations evolve. New linking techniques may also arrive. For example, multi-signature addresses have an unintended negative effect on privacy since the multi-sig structure in a change address can be matched to the sending address even if the keys involved change [46].

To de-anonymize, the adversary must take the further step of linking address clusters to real-world identities. Meiklejohn et al. [75] were successful at identifying clusters belonging to online wallets, merchants, and other service providers since it is easy to learn at least one address associated with such entities by interacting with them. As for identifying regular users, the authors suggest that this may be easy for law enforcement (using subpoena power) since most flows of bitcoins pass through these centralized service providers (who typically require customer identity and keep records). Without such access, however, the adversary is limited precisely due to the centrality of flows—online wallets and other such services mix users’ coins together.

Network de-anonymization. The other major target of de-anonymization efforts is the peer-to-peer network. Nodes leak their IP address when broadcasting transactions. Using an anonymity network is therefore crucial for privacy. However, Biryukov et al. [20] point out a DoS attack to disconnect Tor exit nodes from the Bitcoin network. It remains to be seen if Bitcoin’s P2P layer will evolve to better utilize Tor or if a dedicated anonymity network will be developed. Finally, current SPV implementations provide little anonymity due to the difficulty of privately retrieving the list of transactions that the client is interested in [45].

B. Proposals for improving anonymity

There are three main classes of anonymity proposals. A comparison is provided in Table I with respect to 6 security and deployment properties (with ● meaning a scheme has a property and ◐ indicating it partially does).

¹⁴An alternative payment approach is to use multiple distinct merchant addresses to avoid merges [50], but this is not yet standardized or adopted.

¹⁵One exception is CoinJoin in Section VII-B, which explicitly users multi-input transactions to *increase* anonymity.

TABLE I
COMPARATIVE EVALUATION OF ANONYMITY TECHNIQUES.

Proposal	Class	Security						Deploy.
		Internal Unlinkability	External Unlinkability	Theft Resistance	DoS Resistance	Bitcoin-compatible	# Transactions	
CoinJoin [74]	P2P	●	●				●	1
Shuffle Net [32]	P2P	●	●				●	1
Fair Exchange [12]	P2P		●	●			●	4
CoinShuffle [101]	P2P	●	●	●	◐		●	1
Mixcoin [23]	distr.	◐	●	◐	●		●	2
Blindcoin [115]	distr.	●	●	◐	●		●	4
Zerocoin [76]	altcoin	●	●	●	●			2
Zerocash [15]	altcoin	●	●	●	●			0

Peer-to-peer. In P2P mixing protocols, a set of Bitcoin holders jointly create a series of transactions which (privately) permute ownership of their coins, making each participant anonymous within this set. This process may be repeated with different sets of users to grow the anonymity set.

A straightforward mechanism for achieving this is CoinJoin [74], where a set of users form a single standard Bitcoin transaction with one input from each user and and a fresh output address controlled by each user such that no external party examining the transaction knows which input corresponds to which output (providing external unlinkability). Any user can refuse to sign the transaction if their desired output address is not included, preventing theft but making it vulnerable to DoS by any individual. In vanilla CoinJoin, users announce their output address to the other users (not providing internal unlinkability). This can be addressed through toggling a new Tor circuit or other ad hoc methods, however for robust internal unlinkability, a cryptographic protocol should be used (e.g., CoinShuffle below).

Two earlier proposals offer similar properties to CoinJoin, one based on a shuffling network [32] and one based on fair exchange [12]. However, both are limited to two-party mixing making internal unlinkability impossible. To address the difficulty of finding partners for two party mixing protocols, Xim [21] is a decentralized protocol for finding mixing partners using three stages of fees paid to miners to discourage denial of service attacks.

CoinShuffle [101] is an overlay protocol for forming CoinJoin transactions, adding internal unlinkability through a cryptographic mixing protocol between the participants. It partially (◐) prevents DoS by identifying which parties abort.

Distributed mix network. In Mixcoin [23], users sent standard-sized transactions to a third-party mix and receive back the same amount from coins submitted by other users of the same mix. This provides anonymity toward external entities and partial internal anonymity (◐), as the mix will know the linking between users and output but other users will not. Other users also cannot disrupt the protocol. While mixes may steal Bitcoins at any time, cheating mixes can be identified using signed *warrants* (providing partial ◐ theft

resistance). While Mixcoin’s warranties and other features have not been deployed, this is the closest to third-party mixes as are most commonly used in practice.

Blindcoin [115] extends Mixcoin using blinded tokens similar to Chaum’s original e-cash proposal [25]. This prevents an honest-but-curious mix from learning the mapping between inputs and outputs and upgrade to full internal unlinkability, at a cost of two additional transactions to publish and redeem the blinded tokens.

Altcoins with integrated unlinkability. Zerocoin [76] is an altcoin with integrated unlinkability, using a Bitcoin-esque base currency and an anonymous shadow currency called zerocoins. Users transact solely in the base currency, but can cycle the base currency into and out of zerocoins anonymizing it relative to the set of all zerocoins (a much larger anonymity set than the other techniques above). This provides strong unlinkability with no theft or DoS concerns and without relying on any entities other than miners. However, it is not compatible with Bitcoin and must be implemented as an altcoin (or hard fork). PinnocchioCoin [34] is a similar proposal using a different cryptographic construction.

Zerocash [15] is an even stronger proposal for an anonymous altcoin that is effectively impossible to implement as a hard-fork. Zerocash transactions are a special type of zero-knowledge proofs called SNARKs [16] which reveal no information at all about the amount or recipients (except a possible public transaction fee), enabling a completely untraceable ledger in which no information is revealed publicly. SNARKs are a new cryptographic primitive without any real-world deployment to date and require an initial generation of secret parameters by a trusted party; however, recent work has shown this initial setup can be distributed amongst a set of mutually untrusted parties [17].

VIII. EXTENDING BITCOIN’S FUNCTIONALITY

While Bitcoin can be described simply as a digital currency, the power of the scripting language with enforcement by miners makes many other types of functionalities possible between two or more mutually distrusting parties that would otherwise require a trusted intermediary. We use the term *disintermediation* to refer to the general process of designing transactions that remove the need for a trusted intermediary and observe many proposals to do so by creatively applying or extending Bitcoin’s transaction semantics.

A. Disintermediation with Bitcoin today

The extent to which Bitcoin is an extensible platform is often overstated. The scripting language remains highly constrained. However, many protocols have been designed for disintermediation which can be realized with Bitcoin’s current transaction semantics. We identify three general disintermediation strategies:

Atomicity. In many cases, a desired security property can be enforced directly using functionality provided by the blockchain and the fact that transactions can be *atomic*, being invalid until multiple parties sign. CoinJoin [74] is a simple

example, with no participant’s coins swapped until all parties sign. Another example is Hearn’s “serial micropayments” protocol [51], which makes efficient use of an out-of-band channel to allow one party to authorize a nearly-continuous slow release of funds (e.g., a fraction of a penny per second) in exchange for some service such as Internet access. At any time, either party can abort the protocol by refusing to sign any more transactions, at which point the flow of payment will be complete and only one transaction needs posting to the blockchain. Another clever protocol is Nolan’s atomic cross-chain exchange protocol, which actually allows users to swap currency between two altcoins with two linked transactions and atomic security [87].

Collateral. In other cases, when a desired security property cannot be enforced directly, Bitcoin can provide an acceptable remedy by posting a deposit or bond which is only refunded in the case of correct behavior. This approach is exemplified by the multi-player lottery protocol from Andrychowicz et al. [5]. Each of N parties places a \$1 bet, and one party (chosen at random) walks away with $\$N$. In order to guarantee that a cheating player doesn’t spoil the game by learning the outcome first and selectively aborting the protocol, every player must deposit $\$N^2$. If any participant aborts the protocol they forfeit their deposit, which is used to compensate the others to the maximum amount they could have won. This approach is not limited to lotteries, but in fact can provide a notion of fairness for arbitrary multiparty computations [18].

Auditability. Even if Bitcoin is not used to apply an immediate remedy against a dishonest party, it can still play a crucial role in providing evidence that incriminates the dishonest party. One example is the green addresses technique [53]: here, a payment processor with a well-known public key pledges never to sign an invalid or conflicting transaction. A user who receives a transaction from a green address may accept it (i.e., make an irrevocable decision) before waiting for it to be included in blocks. If at some point the transaction is preempted by a conflicting transaction published in the blockchain, the user obtains easily checkable evidence that the server defaulted. A similar technique is used in Mixcoin [23] to obtain evidence if an anonymity mix has cheated.

B. Cost of disintermediation

Disintermediation usually comes at a cost. Bitcoin’s transaction fee mechanism ensures users pay miners in exchange for enforcing their contracts. The fee mechanism is not perfect; for the time being, transaction fees are optional, and the amount of miner income from fees pales in comparison to the fixed “block creation” subsidy. Default miner policy is currently to assign fees based mainly on transaction size.

Costs to the Bitcoin network. First, since each transaction is propagated to every node in the network, an immediate communication cost is incurred proportional to the number and size of each transaction. Second, nodes (by default) store *every* transaction in history on disk and serve them to new nodes who join the network. Transactions whose outputs have already been spent are no longer needed for validation, and can

TABLE II
PERFORMANCE COMPARISON OF PROTOCOLS USING BITCOIN AS A COMPONENT OR PLATFORM. FOR READABILITY, CELLS THAT ARE EMPTY ARE IMPLICITLY $O(1)$. THE MEANING OF VARIABLES USED IN EACH PROTOCOL IS DEFINED IN THE TEXT.

Protocol	Model	Role	Cost to Parties				Cost on Bitcoin				
			Comp.	State	Net	Collateral	Time	# Trans.	Comp.	State	Net
SPV Payment [85]	Direct	Buyer/Seller					k				
Green Addresses [53]	Auditable	Buyer/Seller Auditors	$O(1)$	$O(1)$		$O(1)$	$O(1)$				
Micropayments [51]	Collateral	Buyer/Seller	$O(p)$		$O(p)$	\bar{p}	$k + O(\bar{p})$				
Lottery [5]	Collateral	Each Party	$O(n)$		$O(n)$	$O(n)$	$2k$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Lottery [18]	Collateral	Each Party	$O(n)$		$O(n)$	$O(n)$	$(n+1)k$	$O(1)$	$O(n)$	$O(1)$	$O(n)$
CoinJoin [74]	Direct	Buyer/Seller	$O(n)$		$O(n)$		k	1			
Mixcoin [23]	Auditable	Buyer/Seller	$O(n)$		$O(n)$		k	1			
Cross-chain exch. [87]	Collateral	Buyer/Seller		1	$2k$	1					

be pruned or moved to archival storage; however, the unspent transaction outputs must be stored in a readily accessible index. Hence, the size and length of time a transaction’s outputs remain unspent, rather than just the size of the transaction, determine the cost imposed on the network. Third, computation is required to validate new transactions. In the worst case, the computation depends on the maximum effort required to reject an *invalid* transaction. Since the Bitcoin scripting language has no looping/iterating instructions, the worst-case validation cost is typically proportional to the size of the transaction output.

Costs to the participants. First, parties may or may not need to perform the full duties of a Bitcoin node (i.e., receiving and processing every transaction). For example, a user that receives a transaction output and must spend it later needs only to maintain his private key; however, an auditor that waits for a message encoded in the blockchain may need to wait for every transaction. Second, for protocols involving collateral-based enforcement, a party may need to make a capital deposit that is returned at the end of the protocol. Third, a party often needs to wait for a transaction to appear “settled” before proceeding; this amount of time depends on the parameters of the system and the user’s desired security.

In Table II we evaluate the asymptotic performance characteristics of a variety of disintermediated blockchain protocols. For protocols that involve heterogeneous roles, we break down the costs to each role on a separate row. We use n for the number of participants in the protocol and k to denote the amount of time needed to wait for a transaction to be confirmed (typically 6 blocks in practice).

For incremental micropayment transactions [51], \bar{p} denotes a maximum amount of money the user may wish to pay over the course of a transaction, while p denotes the total number ultimately paid. Note that for the cross-chain exchange protocol [87], n is limited to two parties.

C. Bitcoin as a data store

An alternate approach to utilizing Bitcoin is to simply use it as global append-only log into which anybody can write data.

Secure timestamping. Because the blockchain is (modulo forks) append-only, it can be used immediately as a secure timestamping service [29], which is useful in a variety of security protocols. Arbitrary data can be written into the blockchain through several mechanisms—the community prefers the use of a small provably unspendable script which allows data in an unused variable.¹⁶ Multiple services collect data from users and publish a Merkle root to the blockchain, allowing anybody to timestamp arbitrary data.

Digital tokens: Colored Coins. Because data can be written into individual transactions, it is possible to mark certain transactions with a “color.” This enables a protocol called Colored Coins [100] which defines a set of rules (not enforced by miners) to transfer color from input transactions to output transactions. Coins may initially be colored by including a special signature from any authority trusted to issue color for some application. This allows the creation of arbitrary tokens which can be traded for each other or for ordinary uncolored bitcoins. This uses the history-tracking functionality of the blockchain as a feature. In general, it has been observed that every transaction output has a unique history of ancestors which may be meaningful to different users, meaning that in the long run bitcoins are not guaranteed to be fungible [84].

Colored coins have been proposed for many applications, such as trading stocks or property rights. Because Bitcoin miners do not enforce the rules of the colored coins protocol, validating a transaction’s color requires scanning the blockchain for all ancestor transactions (thwarting SPV).

Overlay protocols: Mastercoin. A more flexible approach is to use Bitcoin as a consensus mechanism which holds arbitrary data, but implements a completely different transaction-based system within that data. Two prominent such systems are Counterparty and Mastercoin [120], which define a large number of additional transaction types for trading digital assets and contracts. Note that this design removes the transaction validity property that Bitcoin’s consensus mechanism normally enforces. Thus invalid overlay transactions may be published

¹⁶Proof-of-burn is also a solution, but this is not provably unspendable and so it is discouraged by miners.

and have to be ignored. SPV security is impossible as users must validate the entire overlay transaction history.

D. Extending Bitcoin's transaction semantics.

The Bitcoin scripting language is deliberately restrictive; in fact, the original source contains the makings of a much more versatile language, but most of the opcodes are marked as unusable. There are many plausible new opcodes to add to the scripting language, such as new cryptographic primitives, although any new op-code is a hard-forking change.

Mostly orthogonal to the transaction semantics are a variety of mechanisms for computing them; we defer discussion of these to Appendix X.

Turing-complete scripting. Taken to the limit, we may wish for a fully general-purpose policy language, removing the need to gradually add specific opcodes to enable new functionality. This approach is taken by the altcoin Ethereum [122], which provides both a Turing-complete scripting language and a random access datastore. It remains unclear how Ethereum will play out in practice, and designing a practical cryptocurrency with a Turing-complete scripting language remains an interesting open problem.

Application-specific primitives. Many interesting application-specific primitives have been proposed for use with Bitcoin (after a hard-fork) or, more likely in new altcoins. We will not attempt a complete survey here but will discuss several interesting examples. Perhaps the best known example is Namecoin [70] (the first altcoin), which adds logic for mapping strings to values and transferring control over those strings, allowing the system to function as a decentralized naming system. Another proposal is decentralized prediction markets and order books, enabling the purchase and trade of shares in a future event [28].

IX. CONCLUDING REMARKS

Our extensive analysis of Bitcoin based on both the academic and (vast, fragmented) online literature shows how second-generation cryptocurrencies have led to a renaissance of new ideas, addressing important and challenging security problems. Innovation has not been limited to new cryptocurrency protocol designs, but has touched many areas of computer security, distributed systems, hardware design and economics.

Although our scientific knowledge has grown considerably, our understanding is often still lacking. A simple fact demonstrates this: given the chance to design a currency system from scratch, it is unclear what significant deviations from Bitcoin would be desirable or what effects they would have in practice. This is not to say Bitcoin is flawless, as its many design quirks show. There are also several areas, such as anonymity, in which greatly enhanced designs have been proposed. Yet no altcoin has seriously challenged it for market share despite their freedom to improve on the initial design.

Unfortunately, it remains difficult to assess the extent to which Bitcoin's success is due to its specific design choices, as opposed to its first-mover advantage. Our systematization

shows many dimensions in which it is not clear if any alternative proposals are actually "better" or even what "better" should mean. For the consensus protocol, we proposed a standard definition for stability, yet the literature does not provide adequate tools to assess under which economic and social assumptions Bitcoin actually guarantees them. We also do not yet know if it is possible to design an alternate decentralized consensus system which can match Bitcoin's stability in practice while offering better performance or efficiency. For designing new disintermediated protocols, it is not clear what the right approach is to allow expanding Bitcoin's functionality without upsetting its decentralized nature.

Compared to many other areas of security research, Bitcoin is a rare case where practice seems to be ahead of theory. We consider that a tremendous opportunity for the research community to tackle the many open questions about Bitcoin which we have laid out.

ACKNOWLEDGMENTS

The authors would like to thank the following colleagues for feedback on drafts of this paper: Sergio Domain-Lerner, Luke Valenta, Albert Szmigielski, as well as the anonymous reviewers at IEEE Security & Privacy and members of the Bitcoin developer community. Joseph Bonneau is supported by a Secure Usability Fellowship.

REFERENCES

- [1] G. Andresen. March 2013 Chain Fork Post-Mortem. BIP 50.
- [2] G. Andresen. Pay to Script Hash. BIP 16, 1 2012.
- [3] G. Andresen. Blocksize Economics. bitcoinfoundation.org, October 2014.
- [4] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating User Privacy in Bitcoin. In *Financial Cryptography*, 2013.
- [5] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure Multiparty Computations on Bitcoin. In *IEEE Symposium on Security and Privacy*, 2014.
- [6] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged Byzantine impostors. Technical report, Yale, 2005.
- [7] M. Babaiouf, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and Red Balloons. In *EC*, pages 56–73. ACM, 2012.
- [8] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. Enabling blockchain innovations with pegged sidechains, 2014.
- [9] A. Back et al. Hashcash-a denial of service counter-measure, 2002.
- [10] L. Bahack. Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft). Technical Report abs/1312.7013, CoRR, 2013.
- [11] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with Bitcoins. In *IEEE P2P*, pages 1–5. IEEE, 2013.
- [12] S. Barber, X. Boyen, E. Shi, , and E. Uzun. Bitter to Better—How to Make Bitcoin a Better Currency. In *Financial Cryptography*, 2012.
- [13] J. Becker, D. Breuker, T. Heide, J. Holler, H. Rauer, and R. Böhme. Can We Afford Integrity by Proof-of-Work? Scenarios Inspired by the Bitcoin Currency. In *WEIS*, 2012.
- [14] M. Belenkiy. E-Cash. In *Handbook of Financial Cryptography and Security*. CRC, 2011.
- [15] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *IEEE Symposium on Security and Privacy*. IEEE, 2014.
- [16] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *CRYPTO*, 2013.
- [17] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *IEEE Symposium on Security and Privacy*, 2015.
- [18] I. Bentov and R. Kumaresan. How to Use Bitcoin to Design Fair Protocols. In *CRYPTO*, 2014.
- [19] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of Activity: Extending Bitcoin’s Proof of Work via Proof of Stake. Cryptology ePrint Archive, Report 2014/452, 2014.
- [20] A. Biryukov and I. Pustogarov. Bitcoin over Tor isn’t a good idea. In *IEEE Symposium on Security and Privacy*, 2015.
- [21] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore. Sybil-Resistant Mixing for Bitcoin. In *WPES’14: Workshop on Privacy in the Electronic Society*, 2014.
- [22] J. Bonneau and P. ECKERSLEY. Agile Tokens, 2014.
- [23] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for Bitcoin with accountable mixes. In *Financial Cryptography*, March 2014.
- [24] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT*, 2005.
- [25] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.
- [26] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, 1990.
- [27] N. Christin. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. In *WWW*, 2013.
- [28] J. Clark, J. Bonneau, E. W. Felten, J. A. Kroll, A. Miller, and A. Narayanan. On Decentralizing Prediction Markets and Order Books. In *WEIS*, 2014.
- [29] J. Clark and A. Essex. CommitCoin: carbon dating commitments with Bitcoin. In *Financial Cryptography*, 2012.
- [30] M. Corallo. High-speed Bitcoin Relay Network, November 2013.
- [31] N. T. Courtois. On the longest chain rule and programmed self-destruction of crypto currencies. *arXiv preprint arXiv:1405.0534*, 2014.
- [32] O. Coutu. Decentralized Mixers in Bitcoin. *Bitcoin Conference*, 2013.
- [33] W. Dai. b-money. www.weidai.com/bmoney.txt, 1998.
- [34] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno. Pinocchio Coin: building Zerocoin from a succinct pairing-based proof system. In *PETShop*, 2013.
- [35] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P*, pages 1–10. IEEE, 2013.
- [36] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí. The Bitcoin P2P network. In *BITCOIN’14: The First Workshop on Bitcoin Research*, Jan. 2014.
- [37] dree12. List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses. bitcointalk.org, August 2014.
- [38] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, 1992.
- [39] S. Eskandari, D. Barrera, E. Stobert, and J. Clark. A first look at the usability of bitcoin key management. *Workshop on Usable Security (USEC)*, 2015.
- [40] I. Eyal. The Miner’s Dilemma. In *IEEE Symposium on Security and Privacy*, 2015.
- [41] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, March 2014.
- [42] Freicoín - easy-to-use demurrage currency. <http://freico.in/>.
- [43] J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. Cryptology ePrint Archive, Report 2014/765, 2014.
- [44] L. Garber. Government Officials Disrupt Two Major Cyberattack Systems. *IEEE Computer*, July 2014.
- [45] A. Gervais, G. O. Karame, D. Gruber, and S. Capkun. On the Privacy Provisions of Bloom Filters in Lightweight Bitcoin clients. In *ACSAC*, 2015.
- [46] S. Goldfeder. New research: Better wallet security for Bitcoin. http://www.cs.princeton.edu/~stevenag/bitcoin_threshold_signatures.pdf, March 2014.
- [47] D. M. Goldschlag and S. G. Stubblebine. Publicly Verifiable Lotteries: Applications of Delaying Functions. In *Financial Cryptography*, 1998.
- [48] R. Grinberg. Bitcoin: An Innovative Alternative Digital Currency, November 2011.
- [49] M. Hearn. Dan Kaminsky’s thoughts on scalability. bitcointalk.org, 2011.
- [50] M. Hearn. Merge-Avoidance: a note on privacy-enhancing techniques in the Bitcoin protocol. medium.com, 2013.
- [51] M. Hearn. Rapidly-adjusted (micro)payments to a pre-determined party. bitcointalk.org, 2013.
- [52] D. Y. Huang, H. Dharmdasani, S. Meiklejohn, V. Dave, C. Grier, D. McCoy, S. Savage, N. Weaver, A. C. Snoeren, and K. Levchenko. Botcoin: monetizing stolen cycles. In *NDSS*, 2014.
- [53] jav. Instawallet introduces new approach to instant payment: Green address technique. bitcointalk.org, July 2011.
- [54] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *BITCOIN’14: The First Workshop on Bitcoin Research*, Jan. 2014.
- [55] G. O. Karame, E. Androulaki, and S. Capkun. Double-spending fast payments in Bitcoin. In *CCS*, 2012.
- [56] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Foundations of Computer Science*, 2000.
- [57] S. King. Primecoin: Cryptocurrency with prime number proof-of-work, 2013.
- [58] S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, August 2012.
- [59] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *WEIS*, June 2013.
- [60] P. Krugman. Bitcoin is Evil. *The New York Times*, Dec 2013.
- [61] J. Kwon. TenderMint: Consensus without Mining, August 2014.
- [62] S. M. Larson, C. D. Snow, M. Shirts, et al. Folding@ Home and Genome@ Home: Using distributed computing to tackle previously intractable problems in computational biology. Technical report, arXiv preprint, 2002.
- [63] A. Laszka, B. Johnson, and J. Grossklags. When Bitcoin Mining Pools Run Dry: A Game-Theoretic Analysis of the Long-Term Impact of Attacks Between Mining Pools. In *BITCOIN’15: The Second Workshop on Bitcoin Research*, 2015.
- [64] B. Laurie. An Efficient Distributed Currency, 2011.
- [65] B. Laurie. Decentralised currencies are probably impossible (but let’s at least make them efficient), 2011.
- [66] B. Laurie and R. Clayton. Proof-of-work proves not to work. In *WEIS*, volume 2004, 2004.
- [67] T. B. Lee. Major glitch in Bitcoin network sparks sell-off; price

- temporarily falls 23%. *Ars Technica*, March 2013.
- [68] S. D. Lerner. The Private Automatic Miner Backbone Protocol (PAMBA), April 2014.
- [69] P. Litke and J. Stewart. Cryptocurrency-stealing malware landscape. Technical report, Dell SecureWorks Counter Threat Unit, 2014.
- [70] A. Loibl. Namecoin. namecoin.info, 2014.
- [71] makomk. [dead] coiledcoin - yet another cryptocurrency, but with op_eval! bitcointalk.org.
- [72] J. Matonis. The bitcoin mining arms race: Ghash.io and the 51% issue, July 2014.
- [73] G. Maxwell. Merkle tree of open transactions for lite mode? bitcointalk.org, June 2011.
- [74] G. Maxwell. CoinJoin: Bitcoin privacy for the real world. bitcointalk.org, August 2013.
- [75] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *IMC*, 2013.
- [76] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.
- [77] A. Miller. Feather-forks: enforcing a blacklist with sub-50% hash power. bitcointalk.org, October 2013.
- [78] A. Miller, M. Hicks, J. Katz, and E. Shi. Authenticated data structures, generically. In *POPL*, 2014.
- [79] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing Bitcoin Work for Data Preservation. In *IEEE Symposium on Security and Privacy*, May 2014.
- [80] A. Miller and J. J. LaViola Jr. Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin, 2014.
- [81] A. Miller, E. Shi, A. Kosba, and J. Katz. Nonoutsourcable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions (preprint), 2014.
- [82] T. Moore and N. Christin. Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk. In *Financial Cryptography*, 2013.
- [83] M. Möser and R. Böhme. Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees. In *BITCOIN'15: The Second Workshop on Bitcoin Research*, 2015.
- [84] M. Möser, R. Böhme, and D. Breuker. Towards risk scoring of bitcoin transactions. In *BITCOIN*, 2014.
- [85] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. <http://bitcoin.org/bitcoin.pdf>, 2009.
- [86] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2), 1951.
- [87] T. Nolan. Alt chains and atomic transfers. bitcointalk.org, May 2013.
- [88] T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology—CRYPTO'91*, pages 324–337. Springer, 1992.
- [89] M. Okun. Agreement among unacquainted Byzantine generals. In *Distributed Computing*, pages 499–500. Springer, 2005.
- [90] M. Palatinus. Stratum mining protocol - ASIC ready. <https://mining.bitcoin.cz/stratum-mining>, September 2012.
- [91] R. Parhonyi. Micropayment Systems. In *Handbook of Financial Cryptography and Security*. CRC, 2011.
- [92] C. Percival and S. Josefsson. The scrypt Password-Based Key Derivation Function, 2012.
- [93] A. Poelstra. Distributed Consensus from Proof of Stake is Impossible, May 2014.
- [94] QuantumMechanic. Proof of stake instead of proof of work. bitcointalk.org, July 2011.
- [95] A. Rapoport and A. M. Chammah. The game of chicken. *American Behavioral Scientist*, 10(3):10–28, 1966.
- [96] R. L. Rivest. Peppercoin micropayments. In *Financial Cryptography*, 2004.
- [97] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. In *Security Protocols Workshop*, 1997.
- [98] S. Ro. A Bloomberg TV Host Gifted Bitcoin On Air And It Immediately Got Stolen. *Business Insider*, December 2013.
- [99] M. Rosenfeld. Analysis of Bitcoin Pooled Mining Reward Systems. Technical report, CoRR, 2011.
- [100] M. Rosenfeld. Overview of Colored Coins, 2012.
- [101] T. Ruffing, P. Moreno-Sanchez, and A. Kate. CoinShuffle: Practical decentralized coin mixing for Bitcoin. In *ESORICS*, 2014.
- [102] T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *CRYPTO*, 1999.
- [103] T. Sander, A. Ta-Shma, and M. Yung. Blind, auditable membership proofs. In *Financial Cryptography*, 2001.
- [104] B. Schoenmakers. Security aspects of the Ecash™ payment system. *State of the Art in Applied Cryptography*, 1998.
- [105] D. Schwartz, N. Youngs, and A. Britto. The Ripple Protocol Consensus Algorithm, September 2014.
- [106] SEC vs Shavers. <https://www.sec.gov/litigation/complaints/2013/comp-pr2013-132.pdf>, 2013.
- [107] M. Sirbu and J. D. Tygar. NetBill: An internet commerce system optimized for network-delivered services. *Personal Communications, IEEE*, 2(4):34–39, 1995.
- [108] E. G. Sizer and I. Eyal. How to Disincentivize Large Bitcoin Mining Pools, June 2014.
- [109] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing fast money grows on trees. *Not Chains*, 2013.
- [110] I. Stewart. Proof of burn. bitcointalk.org, December 2012.
- [111] N. Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.
- [112] M. B. Taylor. Bitcoin and the age of bespoke Silicon. In *CASES*, 2013.
- [113] P. Todd, G. Maxwell, and O. Andreev. Reducing UTXO: users send parent transactions with their merkle branches. bitcointalk.org, October 2013.
- [114] J. Tromp. Cuckoo Cycle: a memory-hard proof-of-work system. In *BITCOIN'15: The Second Workshop on Bitcoin Research*, 2015.
- [115] L. Valenta and B. Rowan. Blindcoin: Blinded, Accountable Mixes for Bitcoin. In *BITCOIN'15: The Second Workshop on Bitcoin Research*, 2015.
- [116] M. Vasek, M. Thornton, and T. Moore. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *BITCOIN'14: The First Workshop on Bitcoin Research*, Jan. 2014.
- [117] V. Vishnumurthy, S. Chandrakumar, and E. G. Sizer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*, volume 35, 2003.
- [118] F. Voight. p2pool: Decentralized, DOS-resistant, hop-proof pool. <https://bitcointalk.org/index.php?topic=18313.0>, June 2011.
- [119] J. Von Neumann and O. Morgenstern. *Theory of games and economic behavior (2d rev)*. Princeton University Press, 1947.
- [120] J. R. Willett. MasterCoin Complete Specification, v1.1, 2013.
- [121] wizkid057. Re: [6600Th] Eligius: 0% Fee BTC, 105% PPS NMC, No registration, CPPSRB (New Thread). bitcointalk.org, June 2014.
- [122] G. Wood. Ethereum: A secure decentralized transaction ledger. <http://gavwood.com/paper.pdf>, 2014.

X. TRANSACTION VALIDATION COSTS

In the simplest model, system stability follows from the assumption that honest nodes only accept blocks containing valid transactions, where *validity* is defined as a function over the transaction history. In the case of Bitcoin, a transaction is valid only if its input coins (i.e., transaction outputs) have not already been spent. Let N denote the total number of transaction outputs processed at some point in time, and let $M < N$ denote the number of active *unspent* coins.

By what process do honest nodes efficiently determine whether or not a transaction is valid? A naïve approach would be for each honest node to store an entire copy of the transaction history so far, and to check each new transaction by directly computing the validation function. This approach has two immediate drawbacks. First, the cost of storing the transaction history — especially in a readily-accessible rather than archival way — grows without bound over time. Second, the validation function can be expensive to compute directly; validating a Bitcoin transaction could require traversing the entire $O(N)$ history to check each input has not been spent.

To overcome the drawbacks of the naïve approach, Bitcoin nodes maintain a *statefile* that contains indexed data to ef-

TABLE III
ASYMPTOTIC PERFORMANCE TRADEOFFS FOR DIFFERENT DATA STRUCTURES FOR VALIDATING TRANSACTION RULES

	Verifier Cost			Prover Cost		
	Comp.	State	Network	Comp.	State	Network
Naïve	$O(N)$	$O(N)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Bitcoin (store UTXO)	$O(\log M)$	$O(M)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
UTXO Tree ADS [73], [78]	$O(\log M)$	$O(1)$	$O(\log M)$	$O(\log N)$	$O(M)$	$O(N \log M)$
TXO Tree ADS [113]	$O(\log N)$	$O(1)$	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(N \log N)$
SNARKcoin	$O(1)$	$O(1)$	$O(1)$	$O(\text{polylog}N)$	$O(\text{polylog}N)$	$O(N \log N)$

ficiently check transactions.¹⁷ The statefile is a function of the transaction history, and can be incrementally updated each time new transactions are added to the history. In particular, the statefile in Bitcoin contains a $O(M)$ -size table of *unspent transaction outputs* (called the *UTXO set*), indexed for retrieval by transaction hash in $O(\log M)$ time. Other transaction validation rulesets call for different statefiles: for example, in Namecoin, the statefile contains the current mapping of domain names to IP addresses.

The statefile approach has the drawback that honest nodes still maintain copies of a potentially large and growing database. To avoid this, Maxwell [73] proposed replacing the statefile with a hash-based authenticated data structure (UTXO Tree ADS). ADS protocols (e.g., Merkle trees) are protocols allowing a verifier (in this case, an honest node) to outsource the storage of a datastructure to an untrusted prover (e.g., other peers on the network). The verifier stores just a constant sized digest of the data structure; in order to query the data structure, the untrusted prover generates a Verification Object (VO) that the verifier can check. Miller et al. [78] developed a generic programming language for ADS protocols with this application as a case study. The “prover” in an ADS protocol need not be a single entity that stores the entire data structure; with this in mind, Todd, Maxwell, and Andreev [113] proposed an arrangement wherein each client is responsible for maintaining a VO for just transactions they wish to make in the future. In particular, they propose using a data structure that has the desirable property that the VO for a transaction can be incrementally maintained without use of the additional state. However, this data structure has a larger size ($O(N)$ rather than $O(M)$) since it includes both spent and unspent transactions outputs; hence we call it a TXO Tree ADS.

Finally, we observe that the verifier cost in any ADS protocol can be reduced (to a constant) using succinct non-interactive argument systems (SNARKs), though this comes at an additional cost to the prover (and requires an initial setup procedure to be performed by a trusted party); we use the name SNARKcoin to denote such a system.¹⁸

We summarize the performance tradeoffs of the transaction validation protocols discussed above in Table III. For the sake of this evaluation, we concern ourselves only with validation rules of a restricted form, namely rules that involve dictionary lookup by a key; this model suffices for applications ranging from Bitcoin and Ethereum, and given the generic nature of ADS protocols we believe this is without loss of generality. We also note that there are other possible arrangements, including combinations of the ones we’ve discussed. For example, when using an ADS as in [78], [113], we may have a heterogeneous network where clients send their transactions to honest “full” nodes storing the entire $O(M)$ state file, while other honest “lite” nodes verify the VOs generated by the full nodes.

¹⁷Bitcoin nodes, by default, *also* maintain the transaction history, even though this is not needed for transaction validation; such nodes are called *archival*.

¹⁸Zerocash, and PinocchioCoin [34] also use (the zero-knowledge variant of) SNARKs in a related way. However, their goal is to provide anonymity rather than to reduce verifier cost to a constant. Indeed transaction validation in their schemes requires additionally checking a plaintext table of spent serial numbers.