# Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies

Florian Tschorsch Björn Scheuermann

Humboldt University of Berlin
{tschorsch, scheuermann}@informatik.hu-berlin.de

## ABSTRACT

Besides attracting a billion dollar economy, Bitcoin revolutionized the field of digital currencies and influenced many adjacent areas. This also induced significant scientific interest. In this survey, we unroll and structure the manyfold results and research directions. We start by introducing the Bitcoin protocol and its building blocks. From there we continue to explore the design space by discussing existing contributions and results. In the process, we deduce the fundamental structures and insights at the core of the Bitcoin protocol and its applications. As we show and discuss, many key ideas are likewise applicable in various other fields, so that their impact reaches far beyond Bitcoin itself.

## 1. INTRODUCTION

Over the past decades, the Internet has witnessed the advent of many bottom-up, grassroots applications which solve problems in a cooperative, distributed manner. A number of these community-driven, non-commercial systems has become well-known and widespread; examples include anonymous communication [Chaum 1981], PGP [Zimmermann 1995], Hashcash [Back 2002], and BitTorrent [Cohen 2003]. If fact, practically applicable solutions have often become available soon after the idea for a certain application had first been conceived. Digital money is an exception from this rule: from the early 1980s the vision of digital money had been around—but it took more than a quarter of a century before a fully distributed solution became reality.

The early attempts to build digital currencies, as described in [Chaum 1982, Law et al. 1996], require a central authority—that is, a bank. Approaches like b-money [Dai 1998], RPOW [Finney 2004], and bit gold [Szabo 2005] later came up with the idea to interpret the solution to a cryptographic puzzle—a proof of work—as something valuable. They compared it to a piece of precious metal or a minted coin. This way everybody could become a "digital gold digger", mining money independently from a bank, but still requiring a central instance to maintain ownership records.

In order to completely eliminate the bank, the ledger which accounts for the ownership of coins must also be distributed. However, a fundamental and inherent risk of digital currencies in general—and a distributed currency in particular—is the ability to double spend coins. Since digital copies are trivial, someone could issue two transactions in parallel, transferring the same coin to different recipients. In an online and centralized scenario, the bank is able to detect and prevent the attempt. Accomplishing the same in a distributed setting is far from trivial. The distribution of information and the problem of mutual agreement on a consistent state is a challenge, especially in the presence of selfish and/or malicious participants. It boils down to the Byzantine Generals problem [Lamport et al. 1982]. This insight [Szabo 2003] pushed the idea to employ quorum systems [Szabo 1998]. Quorum systems, as described in [Malkhi and Reiter 1998], accept the possibility of faulty information and the existence of malicious entities in a distributed environment. They introduce the concept of voting. As long as the majority of any chosen subset of peers (quorum) is honest, the correct value is obtained by election. However, the approach is vulnerable to the Sybil attack [Douceur 2002]: a malicious entity could set up many peers which subvert the election and inject faulty information. Furthermore, it ignores the propagation delays in distributed systems and leads to temporary inconsistencies.

These difficulties were finally overcome by the Bitcoin design [Nakamoto 2008a]. In November 2008 it was announced by Satoshi Nakamoto to the Cryptography mailing list [Nakamoto 2008b]. After its deployment in 2009, Bitcoin quickly became viral. Nakamoto remained active until about 2010, before handing over the project. Until now the true identity of Nakamoto remains unknown and is subject to speculation, e. g. whether the name is real or a pseudonym, or if it in fact represents a group of people. That much is certain: Bitcoin cleverly combines existing contributions from decades of research [Merkle 1987, Malkhi and Reiter 1998, Haber and Stornetta 1991, Massias et al. 1999, Back 2002]. Beyond that, however, it also solved fundamental problems in a highly sophisticated, original and practically viable way: it uses a proof of work scheme to limit the number of votes per entity, and thus renders decentralization practical.

Bitcoin miners collect transactions in a block and vary a nonce until one of them finds the solution to a given puzzle. The block including the solution and the transactions are broadcast to all other entities, and the distributed ledger (the *block chain*) is updated. The ownership of coins can be determined by traversing the block chain until the most recent transaction of the respective coin is found. Forks of the block chain due to malicious manipulations or propagation delays are resolved by considering the "longest" fork (including most of the work) as consensus. Thus, Sybil and—at least to some extent—double spending attacks are mitigated by binding additions to the block chain (votes) to proof-of-work contributions. The proof of work also induces a continuous supply of new coins as a reward (and incentive) for miners. All of this does not require a centralized coordinating authority, and practically demonstrates the feasibility of a distributed digital currency.

Early Bitcoin studies gave a preliminary overview of the system's strengths and weaknesses [Barber et al. 2012] and compared them to paper and electronic cash [Drainville 2012]. This survey describes and reflects the state of the art in the area of fully distributed digital currencies. Today, this reaches significantly beyond Bitcoin. Yet, Bitcoin is still by far the most widely known system, and it marks the turning point which accelerated the entire research area. We therefore put Bitcoin at the center of our attention, and arrange related and alternative concepts around it. We will start from Bitcoin's proof-of-work concept, from where we explore the technical background and discuss the implications of the central design decisions. Based on a detailed understanding of these aspects, the research area beyond Bitcoin unfolds, so that we iteratively take alternative approaches into the discussion. For instance, this includes alternatives to the proof-of-work scheme or Bitcoin-like distributed consensus schemes for other applications. Many of these approaches result in new currencies, so called "altcoins", which exist concurrently to Bitcoin.

In order to gain a full technical understanding of Bitcoin as it is used today, scientific literature alone is not sufficient. Many important details can only be found in mailing lists, forum posts, blogs, wikis, and source code—some of them dating back until the 1980s. This survey aims to provide the whole picture. Therefore, we also took these sources into account, to incorporate also those aspects that have previously not been described with scientific rigour. In fact, even though Bitcoin has been recognized controversially and gained media attention due to being used for criminal purposes, from a technical perspective, we perceive a certain beauty in the system, which may not necessarily be visible at a first glance, but which we hope to convey through our description.

The four goals of this survey are: (i) provide an holistic technical perspective on distributed crypto currencies, (ii) explore the design space and expose the implications of certain design decisions, (iii) consolidate and link the broad body of work while distilling the key algorithmic features, and (iv) identify the fundamental ideas which remain independent from specific implementations or temporary idioms of usage.

Likely the closest relative to this article—and worth a pointer in this context—is [Bonneau et al. 2015]. The authors outline the key elements of Bitcoin's design and the existing body of work in a condensed form for readers with substantial prior knowledge in the area. They also motivate further research in the field and discuss research challenges. Here, we address a broader audience by including a tutorial-style introductory part, and we include a more comprehensive selection from the existing literature for a more in-depth overview.

The remainder of this survey is structured as follows. Section 2 introduces the basics of Bitcoin and provides a first outline of central concepts, including proof of work and double spending. In Section 3 we then consider security threads and implications. Bitcoin's backbone, the peer-to-peer network, influences virtually every aspect of the currency; we cover it in Section 4. The subsequent Section 5 discusses the balance between privacy properties and the system's inherent transparency; here, we review many approaches to enhance privacy. In Section 6 and we revisit the proof-of-work scheme and related topics, and in particular discuss alternative approaches. Finally, before concluding this survey in Section 8, Section 7 summarizes our key observations.

# 2. THE BITCOIN PROTOCOL

In this section we will explain the core Bitcoin protocol as originally introduced in [Nakamoto 2008a]. This will pave the ground for more in-depth discussions of specific aspects in subsequent sections. We begin with an abstract and rather simplistic view of a digital currency, which we then iteratively refine. We also sketch the research context of the presented ideas, and, where appropriate, follow the early steps of digital currencies before Bitcoin [Chaum 1982, Law et al. 1996]. However, our discussions are always directed towards the foundations of the Bitcoin protocol and its main idea: the use of proof of work to eliminate the bank and to decentralize and secure the ledger. In particular, this section will successively introduce the basics on mining, the block chain, transactions and scripting.

The Bitcoin developer documentation [Bitcoin dev 2014], the Bitcoin wiki [Bitcoin wiki 2014] and the Bitcoin source code (`github.com/bitcoin/bitcoin`) constituted valuable sources for the aspects discussed in this section. For a tutorial-like explanation of the Bitcoin basics from a slightly different angle, we also refer the reader to the blog post [Nielsen 2013].

## 2.1 The Starting Point: Centralized Digital Currencies

Assume Alice intends to transfer a coin to Bob. In order to do so—in an extremely naive approach—she could generate a digitally signed contract stating "I transfer one coin to Bob" and announce it publicly. Following Bitcoin terminology, such a contract may be called *transaction* (TX). For the moment, we can consider it as a signed contract, which is verifiable using Alice's public key. It is per se not forgery proof, though, because it can be replayed: if a duplicate copy of the contract appears, it cannot be decided whether Alice tries to fool Bob, whether she (perfectly honestly) aims to transfer a second coin to Bob, or whether Bob performs a replay attack in order to claim multiple coins from Alice's account.

Obviously, to solve such ambiguities, uniquely identifiable coins are necessary. This could be achieved by introducing serial numbers—but where do they come from? We need a trusted source which issues the serials. In a centralized scenario this is what is generically called a *bank*: the bank issues coins with unique serial numbers and maintains a ledger including all ownerships, i. e., the mapping between user accounts and serial numbers.

Transferring a coin would then consist of Alice signing and announcing a transaction of the following form: "I transfer coin #1210 to Bob". Bob verifies the ownership of coin #1210 by consulting the bank. If the transaction is valid and Bob accepts the transaction, the bank updates its ledger. In this moment the owner of the coin changes from Alice to Bob.

This simple, centralized digital currency exemplifies the basic design of the banking model. In fact, it resembles the classical baseline of online electronic payment protocols [Chaum 1982, Law et al. 1996] (even though, of course, they include many clever extensions and additional features). An overview of further technologies in this domain can be found in [Asokan et al. 1997].

Bitcoin aims for a much more ambitious solution, though: one which gets rid of the central bank. To this end, mechanisms are needed to create coins in a distributed setting, and to store and manage the ledger in a distributed way. The key challenge is to achieve consensus on existing coins and their ownership without a central instance, and without mutual trust relationships between participants.

## 2.2 Proof of Work: Decentralizing the Currency

So, how can we eliminate the central bank? Bitcoin solves this in a very pragmatic way: in a sense, everyone is the bank. That is, every participant keeps a copy of the record which would classically be stored at the central bank. We can consider it a distributed ledger reflecting all transactions and ownerships. In Bitcoin, the so-called *block chain* takes the role of this distributed ledger.

However, distributed storage of multiple copies of the block chain opens up new possibilities for Alice to cheat. In particular, Alice could issue two separate transactions to two *different* receivers (say, Bob and Charlie), transferring the *same* coin. This is called *double spending*. If Bob and Charlie verified and accepted the transactions independently (based on their respective local copy of the block chain), this would drive the block chain into an inconsistent state. If Bob accepts the transaction and tells everyone else about it *before* Charlie accepts it, though, Charlie would be able to identify the transaction as a double spending attempt. Thus, under the assumption of a synchronous and unjammable broadcast channel, a simple, synchronized distributed ledger is viable. This is the essence of the simplified B-Money proposal [Dai 1998]. This assumption does not hold in practice, though. Therefore, we have to deal with the undesirable period between issuing a transaction and having everyone informed about it—a prototypical distributed consensus problem.

Bitcoin addresses this problem by, in a sense, letting the entire network verify the legitimacy of the transactions, so that double spending will be noticed by other participants. If and only if a majority of the participants agrees on the existence and legitimacy of the transaction, Bob should accept it. However, this, in turn, raises the question of false identities: an adversary could mount a Sybil attack [Douceur 2002]. That is, Alice could set up many instances all confirming the transaction (thus constituting the "majority"), even though it is, in fact, a double spend. Bob would believe them and accept the transaction.

The Bitcoin protocol makes use of *proof of work* to prevent Sybil attacks. Before verifying a transaction and spreading the news about it, participants have to perform some work to proof they are "real" identities. The work consists of a cryptographic puzzle, which artificially increases the computational cost to verify transactions. Thereby, the ability of verification depends on the computing power, and not on the number of (potentially fake) identities. The underlying assumption is that it is much harder to control the majority of the computing power in the system than it is to control the majority of the identities.

Such proof-of-work schemes have (also before Bitcoin) been used in other areas, for instance against denial of service attacks or spam. Likely one of the best-known examples is Hashcash [Back 2002].

New Bitcoin transactions are communicated to all participants in the network. Given they are valid, these transactions are collected to form a so-called *block*. The puzzle used in the proof of work-based distributed validation process consists of calculating a hash of the thus formed block and adjusting a nonce in such a way that the output value is lower than or equal to a certain target value. Once one participant has found such a nonce, the block with the respective nonce will be distributed in the network, and participants will update their local copy of the block chain.

Solving the puzzle is computationally difficult. Bitcoin uses the SHA-256 hash function [Eastlake and Hansen 2011]. Unless the (cryptographic) hash function used for calculating the block hashes is broken, the only fruitful strategy is to try different nonces until a solution is found. Consequently, the difficulty of the puzzle depends on the target value.

If, for instance, the target demands that the (binary) hash begins with 48 zeros, an average of $2^{48}$ attempts are needed before the puzzle is solved. The lower the target (i. e., the more leading zeros are required), the more difficult the puzzle becomes. Given that all network participants aim to solve the puzzle, the chance of being the first to come up with a solution is proportional to the fraction of the total computing power. Sometimes the analogy of "raffle tickets" is used: the number of tickets for a given participant is proportional to her computing power. The total number of tickets in the raffle wheel is proportional to the total computing power in the system. The more tickets in the raffle wheel, the lower are my chances of winning. However, I can increase my chances by buying more raffle tickets, i. e., by increasing my computing power.

For reasons of stability and reasonable waiting times for transaction validation, the target value is adjusted every 2,016 blocks. It is then re-chosen to meet a verification rate of approximately one block every 10 minutes. Thus, on average, every two weeks ($= 2016 \cdot 10\,\text{min}$) the target is recalculated. The new target $T$ is given by

$$T = T_{\text{prev}} \cdot \frac{t_{\text{actual}}}{2016 \cdot 10\,\text{min}} \tag{1}$$

where $T_{\text{prev}}$ is the old target value and $t_{\text{actual}}$ the time span it actually took to generate the last 2,016 blocks. However, the target never changes by more than a factor of four. If 2,016 blocks were generated during a time span shorter than two weeks, this indicates that the overall computing power has increased, so that the proof of work difficulty should also be increased.

In Bitcoin, the term *difficulty* is used in the specific meaning to express how difficult it is to find a hash below a given target. The ratio of the highest possible target and the current target is used as the difficulty measure.

Let us recap this in the context of our example, where Alice still wants to transfer coins to Bob. When Alice broadcasts her transaction, Bob, Charlie and many others receive it. They all verify its legitimacy based on their local copy of the block chain. Subsequently they enqueue it to the pending transactions they have heard of (that is, they add it to the current block). If Charlie intends to spread his "opinion" that the collection of pending transactions is valid, he first needs to solve the puzzle. Let us assume Charlie is the first participant who solves the puzzle, i. e., the first one to find a nonce for which the hash value meets the target. He then broadcasts the block of transactions together with this nonce. Other participants can verify that the nonce is a valid solution, and hence add the new block to their block chain.

At this point, it is considered consensus that Alice's transfer of the coin to Bob is valid, and Bob is the new owner of the respective coin.

But, after all: why should anyone solve this puzzle and waste computation time (thus energy, ergo money) for verifying and certifying other people's transactions? What does Charlie gain from doing so? In order to provide an incentive, the Bitcoin protocol rewards the first participant who provides the proof of work with coins. There are two sources for this reward: *transaction fees* and *mining*. For now, we concentrate on mining and cover transaction fees only briefly. Later on, we get into more details also on the fees.

Mining is the process of adding new blocks to the block chain, because—in addition to securing the ledger—it results in the generation of new coins. Just like precious metals and collectibles, the block has an unforgeable scarcity: recall that parameters are chosen such that there is one successful puzzle solution roughly every ten minutes. This scarcity creates a value, which is backed up by the real-world (computational) resources required to mint it.

Note that mining also provides a controlled and predictable supply of coins, which does not involve a central bank. Bitcoin's precursors [Dai 1998, Finney 2004, Szabo 2005] already made this fundamental observation and incorporated it in their designs.

In Bitcoin, the initial block reward was set to 50 coins (50 BTC). Every 210,000 blocks, that is approximately once every four years ($= 210{,}000 \cdot 10\,\text{min}$), the reward halves. This happened the first time by the end of November 2012[1]. Since then, the reward for a solved proof-of-work puzzle is 25 BTC. The iterative halving will continue until the mining reward drops below $10^{-8}$ BTC. This amount is the minimal unit of Bitcoin, also known as *satoshi*.

This event is predicted for the year 2140. The minting of new coins will then stop and all ever existing coins (approximately $21 \cdot 10^6$) will be in circulation. However, due to the transaction fees an incentive to validate new blocks will still remain. Indeed, since the first halving of the block reward, transaction fees have increased substantially. This trend will most likely continue and provide the necessary incentive. [Kaskaloglu 2014] discusses the relationship of mining rewards and transaction fees.

## 2.3 Block Chain

So far, we gave only an abstract explanation of the block chain and denoted it as the distributed ledger. That still hits the spot—but there is more. We will now take a closer look at the structure of the block chain. In particular, we will turn towards the question how Bitcoin keeps the blocks in order and comes to a system-wide consistent consensus.

To determine the ownership of each coin, a total order of blocks (and thus transactions) is desirable. For this reason, blocks include a pointer to the previously validated block in the chain. This is illustrated in Figure 1. The pointer is implemented by including a hash of the preceding block. Consequently, the block chain has the structure of a linked list. The so-called *block height* is the number of blocks from head to tail. The block proves that a particular transaction must have existed at the time to get into the block. In this sense, Bitcoin implements a distributed variant of a timestamp service along the lines of [Haber and Stornetta 1991].

[1] `https://blockexplorer.com/b/210000`

Because of the continuous mining, the block chain constantly grows. Due to the popularity of Bitcoin in general and gambles such as SatoshiDice (`satoshidice.com`) in particular, the number of transactions has increased enormously. For instance, bets on SatoshiDice result in two transactions: the stake and the payout (which is at least one satoshi). The winner is determined by a pseudo-random number, which is derived from hashing a daily changing secret and information extracted from the transaction. Their transaction volume peaked in June 2012 with about 62,000 daily transactions. As a consequence, this inflates the block size and results in a non-negligible size of the block chain, currently in the order of tens of gigabytes. Furthermore, the high number of transactions increases the effort of the validation procedure itself. In order to keep the size and the computational effort low, Bitcoin offers a simplified payment verification (SPV) [Nakamoto 2008a] based on Merkle trees [Merkle 1987]. It takes the transactions as leaves and builds a hash tree on top. The root of this tree is a hash value including information from all transactions; this root is included in the block header. The hash tree enables the verification of transactions without the need for a complete local copy of all transactions. Since the root is known and secured through the mining process, branches can be loaded on demand from untrusted sources. Tampering with any transaction would result in different hash values and would thus be detected.

Because block validations are calculated in a distributed way through mining, *forks* can occur: independent block validations can be broadcast almost simultaneously, or while the distribution of one validated block is stalled due to propagation delays. In case of a fork, there are two (or more) different versions of the linked list with, potentially, different sets of included transactions. That is, different participants in the system will disagree on the structure of the block chain. Consequently, there might no longer be a consensus on the order of transactions. Hence, ownership is not settled.

Bitcoin solves this issue by a simple, but effective rule: mining is continued on the "longest" locally known fork, that is, the one involving the highest amount of computational effort so far. At some point, miners of one fork will broadcast validations before others. Thus, one of the forks will "overtake" the other and, once it has been propagated, will become the longest fork for all participants. Thereby, distributed consensus is restored.

If, for example, Alice tries to double spend the same coin with Bob and Charlie and shares the two transactions with two separate subsets of miners respectively, the block chain may fork. Eventually, only one fork will survive, that is, the longer fork will be considered the valid block chain. The other fork is then called *orphaned* and the included transactions are nullified. However, valid transactions of one fork may already be part of the other fork, too, or they will be added to the next block of transactions. Assuming Alice's transaction to Bob made it into the valid block chain, the transaction to Charlie will not be considered anymore, because it attempts to double spend coins. Charlie will recognize this situation after the fork is resolved.

Nevertheless, double spending is still possible under certain circumstances. Suppose Alice buys Charlie's car and wants to pay with Bitcoin. Hence she issues and broadcasts a transaction transferring the agreed amount to Charlie. At

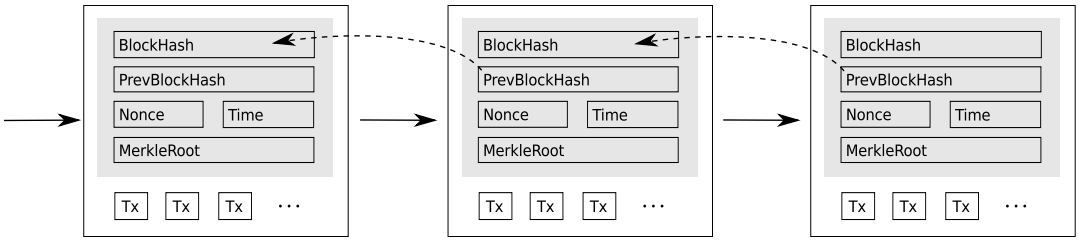



Figure 1: Simplified block chain.

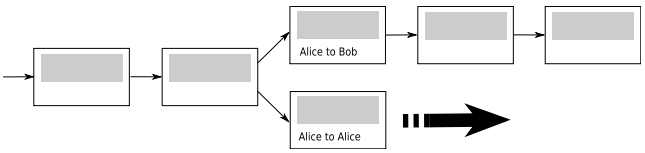



Figure 2: Double spending (race attack).

some point the transaction is included in a block and hooked into the block chain. Eventually the block chain grows, and additional blocks add up as depicted in Figure 2. In return, Charlie hands over the keys to his car. With malicious intentions in mind, Alice creates a conflicting transaction, transferring the coins to another account (e. g., one of her own ones). If she controls enough computing power (more than all honest miners together), she can start a fork and "catch up" with the block chain until her fork becomes longer and thus gets accepted by all others. As a result, she gets her money back. In general, someone who controls more than half of the total computing power can decide which blocks ultimately get accepted in the block chain. Of course, the longer back transactions lie, the more blocks need to be caught up until a fraudulent chain gets accepted. This limits adversaries' possibilities to revise the history of transactions. As a rule of thumb, transactions are commonly considered steady after six consecutive block verifications.

## 2.4 Transactions

Until now we have not exactly stated what "coins" are in the Bitcoin protocol. In fact, coins as such do not exist: there are only transactions, which elaborately assign ownership rights. Thus, the closest actual equivalent to a coin we can think of is the chain of transactions. This will become clear by having a closer look at Bitcoin transactions and how they are composed.

Before receiving coins, Bob needs a virtual wallet consisting at least of a public/private key pair. Bob's Bitcoin *address* is derived from his public key, by hashing it with SHA-256 first and RIPEMD-160 subsequently, prepending a version number, and appending a checksum for error detection. Addresses are base58-encoded to eliminate ambiguous characters. The purpose of Bitcoin addresses is to shorten and obfuscate public keys. In order to receive payments they are not strictly necessary (one could also use the public key or a secret), but they provide a secure and convenient way. It is recommended to avoid key (and thus address) reuse because it harms security and privacy: in particular, it enables comparison-based attacks on signatures [Bos et al. 2013] and tracking of coin flows [Ron and Shamir 2013, Fleder et al. 2013]. Instead, a new key and address should be used for each transaction.

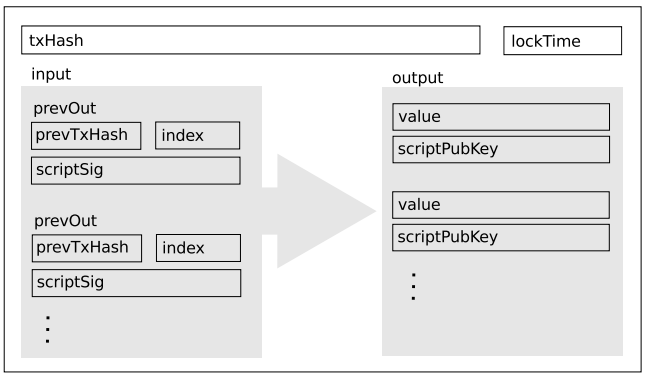



Figure 3: Transaction.

Assume Bob sends his Bitcoin address to Alice. Alice uses her wallet to issue a transaction with Bob's address as the destination. Figure 3 schematically depicts a transaction as it could occur when Alice transfers a coin to Bob.

The key elements of a transaction are a hash value as the transaction identifier (TXID) and a list of *inputs* and *outputs*. Alice uses the input to reference a so far unused output of an earlier transaction. More specifically, `prevTx` is the hash identifying the previous transaction and `index` is the index of the respective output in that transaction. Each output of a transaction can only be used once as an input in the whole block chain. Referencing the same output again is an attempt to spend the same coin twice (double spending) and thus forbidden. Due to this property, each output of a transaction can be categorized either as an *unspent transaction output (UTXO)* if it has not been referenced by a subsequent transaction so far, or as a *spent transaction output (STXO)*.

In the most basic type of transaction, Alice proves that she is able to to mandate over the output referenced on the input side by providing her public key and a signature. For each output, she needs to specify how many coins are to be transferred via this output (`value`) and how to authorize when spending these coins (`scriptKeyPub`); the latter might refer to Bob's bitcoin address as the destination. The Bitcoin transaction system is much more powerful and flexible, though; this will become clear when we cover the concept of scripting.

It is important to note that the transaction input does not specify how many coins are spent. Because each output of a previous transaction can be used only once, inputs necessarily always use all the coins from the referenced output. Since transactions can have multiple inputs and multiple outputs, this does not restrict Bitcoin: Alice can use an additional

output of her transaction to assign part of the coins to her own address, thereby not transferring all the coins from the inputs to others. In this way, Bitcoin implements the idea of *change*.

The sum of all inputs in a standard transaction must be at least as much as the sum of all the outputs. It need not be equal, though: if the input sum is greater than the outputs, implicitly the difference is assigned as transaction fee to the miner validating the block which contains the respective transaction.

The fact that transactions are linked leads us to an inherent and important property of Bitcoin: it is possible to trace transactions back in history. Eventually, doing so will terminate at one out of two possible origins: the *genesis block* or a *coinbase transaction*. Both include special transactions with outputs only.

The genesis block[2] is the origin of the block chain and provides the initial supply of 50 BTC to the system. A coinbase transaction is the much more common origin of a series of transactions. It is the transaction which rewards the miner for validating a block, thereby introducing new coins into the system. Since block chain forks can occur and some blocks will eventually become orphaned, coinbase transactions are locked for at least 100 blocks (i. e., they cannot be spent before 100 subsequent blocks have been verified).

Since coins as such do not exist in Bitcoin, there are also no serial numbers of for coins. The transaction hashes and the reference to the previous transactions take the role of serial numbers as they are used in other digital cash systems. This, finally, also eliminates the need for a bank issuing serial numbers.

## 2.5 Scripts

As indicated before, Bitcoin transactions provide much more flexibility than just the simple coin transfers outlined above. In fact, through *scripting* there is a certain degree of programmability what exactly a transaction does. Scripting is realized by a simple stack-based language. It is intentionally designed not to be Turing complete, so that it is easier to handle and unintended side effects can be avoided.

For the following, bear in mind that a coin is only a generic term for a balance determined by a chain of transactions. Recall that each input of a Bitcoin transaction connects to a given, previous output. Each output in a Bitcoin transaction is described by a script. The operations to be performed, potentially along with constants, constitute the so-called `scriptPubKey`. A script expects a number of "arguments", the `scriptSig`. An input which connects to an output must provide the `scriptSig` for the respective script. The connection is considered valid when the output's script evaluates to `true` given the `scriptSig` provided in the connecting input.

The probably most essential and most common script of all is "Pay-to-PubKeyHash" (P2PKH). Semantically, a transaction employing P2PKH transfers coins from one or more origin addresses to a destination address. The key idea is to have a script at the output which checks whether the connecting input has been signed with the public key "owning" the coins at the output. Script 1 provides the generic P2PKH script template. Bitcoin scripts are processed from left to right. In its `scriptSig`, P2PKH requires a public key (`pubKey`) which hashes to the specified Bitcoin address

[2] `https://blockexplorer.com/b/0`

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash>
  OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>
```

Script 1: "Pay-to-PubKeyHash" (P2PKH) standard script template.

(`pubKeyHash`) and a signature (`sig`) proving the possession of the respective private key.

Consider Alice's transaction to Bob. Alice would substitute the `pubKeyHash` token in Script 1 by Bob's Bitcoin address. She would then include the resulting script as one output script in her transaction to Bob, associated with the value she intends to transfer. If Bob wants to spend the coins again, he needs to provide his public key and his signature in the connecting input's `scriptSig`.

Table 1 shows the script execution and the state of the stack step by step. Here, `pubKeyBob`, `pubKeyBobHash` and `sigBob` denote Bob's public key, the hash of Bob's public key (i. e., Bob's address) and Bob's signature, respectively. First, `scriptSig` (the "arguments") from the input and the `scriptPubKey` (the "code") from the output are concatenated. The first tokens in the concatenated result are `sigBob` and `pubKeyBob`. These are constants values; constants are simply pushed to the stack when they appear in a script. `OP_DUP` duplicates the most recent entry, i. e., `pubKeyBob`, on the stack. `OP_HASH160` hashes the most recent entry twice (SHA-256 and RIPEMD-160) and pushes the result, in this case `pubKeyBobHash`. In the next step, `pubKeyBobHash` (as inserted into the script template by Alice) gets pushed to the stack.

`OP_EQUALVERIFY` verifies the equality of the two topmost stack entries and raises an error if they differ. More generally, the suffix `VERIFY` as in `OP_EQUALVERIFY` indicates a combination of two steps, where the second one is equivalent to `OP_VERIFY`: `OP_VERIFY` takes the topmost element from the stack and marks the transaction invalid if this element is not `true`. This step concludes the first important check: whether the correct public key has been provided. The last remaining operation, `OP_CHECKSIG`, then checks the signature against the public key and pushes `true` if they match. If this final check also passes, the script legitimates Bob to spend the coins: a transaction is valid if nothing fails and the topmost stack entry upon termination is `true`.

Output scripts as discussed so far are created by the originator of the transaction, i. e., the payer. However, it might be desirable for the payment receiver to specify the output script, for instance to ensure long-term security. Of course, after receiving payments, receivers could transfer the coins to themselves with a customized output script. This is not really convenient, though, and it incurs additional transaction fees. Therefore, "Pay-to-ScriptHash" (P2SH) transactions were created and admitted as a standard transaction [Andresen 2012]. P2SH enables the receiver to specify a so-called *redeem script* (`redeemScript`). The hash of the redeem script is transformed into a Bitcoin address-like format [Andresen 2011b] and sent to the originator instead of the recipient's Bitcoin address. The hash is included in a generic output script and can be redeemed as specified in Script 2. In principle, the redeem script can be any script,

| Stack | Script |
|---|---|
| | `sigBob pubKeyBob OP_DUP OP_HASH160 pubKeyBobHash OP_EQUALVERIFY OP_CHECKSIG` |
| `sigBob pubKeyBob` | `OP_DUP OP_HASH160 pubKeyBobHash OP_EQUALVERIFY OP_CHECKSIG` |
| `sigBob pubKeyBob pubKeyBob` | `OP_HASH160 pubKeyBobHash OP_EQUALVERIFY OP_CHECKSIG` |
| `sigBob pubKeyBob pubKeyBobHash` | `pubKeyBobHash OP_EQUALVERIFY OP_CHECKSIG` |
| `sigBob pubKeyBob pubKeyBobHash pubKeyBobHash` | `OP_EQUALVERIFY OP_CHECKSIG` |
| `sigBob pubKeyBob` | `OP_CHECKSIG` |
| `true` | |

Table 1: P2PKH exemplary execution.

```
scriptPubKey: OP_HASH160 <redeemScriptHash>
  OP_EQUAL

scriptSig: [<sig> ...] <redeemScript>
```

Script 2: "Pay-to-ScriptHash" (P2SH) standard script template.

```
scriptPubKey: <m> <pubKey> [<pubkey> ...] <n>
  OP_CHECKMULTISIG

scriptSig: 0 [<sig> ...]
```

Script 3: $m$-of-$n$ multi-signature transaction script template.

but the transaction is considered as a standard transaction only if the redeem script follows one of the standard "pay-to-x" scripts, e. g., a P2PKH. P2SH also supports future development and makes it easier to introduce and roll out new standard transaction types. The idea of P2SH has been generalized in [Gerhardt and Hanke 2012].

Scripting in Bitcoin provides a huge variety of ways to spend coins. For instance, distributed contracts with minimal trust become possible. One building block are $m$-of-$n$ multi-signature transactions [Andresen 2011a], which require $m$ valid out of $n$ possible signatures to redeem a transaction. They are used for (but not limited to) deposits, escrow and dispute mediation. Because of their relevance and to get an impression of Bitcoin's advanced scripting capabilities, we take a somewhat closer look here.

Script 3 depicts the generic script template for an $m$-of-$n$ multi-signature transaction. After pushing the constants to the stack, `OP_CHECKMULTISIG` takes the integer $n$ first (because after pushing, it is the topmost entry), then $n$ `pubKey` items, subsequently the integer $m$, and finally $m$ `sig` items off the stack. Due to a bug in Bitcoin's implementation of `OP_CHECKMULTISIG` (it pops once too often from the stack), it is required to provide one extra value in the `scriptSig`, which is not used. (Due to the requirement of maintaining compatibility, this bug cannot be fixed.) Now, in essence, `OP_CHECKMULTISIG` iterates over public key / signature pairs and executes `OP_CHECKSIG`. Every time a match is found, the script moves on to the next signature or otherwise tries the next public key. As a consequence, the provided signatures in `scriptSig` must be in the order of the appearance of their matching signature in `scriptPubKey`. If the $m$ signatures match, the script pushes `true` to the stack and legitimates the transaction.

The use cases for $m$-of-$n$ multi-signature transactions are manifold. A 2-of-3 multi-signature transaction can, for example, be used to realize customer protection via an independent mediator. In such a constellation, coins are locked and neither the buyer nor the seller alone can claim them. If, however, both agree, the buyer could pass a half-signed transaction over to the seller, who is now able to complete the transaction. In case of a dispute, the mediator can side with one of the participants and clear the situation by providing her signature. Another use case is to secure online wallets. If all funds are held in 2-of-2 multi-signature transactions, where one key is not stored in the online wallet, a thief would not be able to rob a wallet by hacking the online wallet provider.

$m$-of-$n$ scripts can be realized with P2SH scripts and thus can be specified by the receiver(s). Here, the `scriptPubKey` becomes the redeem script. If $n \leq 3$, the transaction is considered a standard transaction.

Even though Bitcoin's scripting has a limited instruction set, the possibilities are manifold. The above mentioned examples can be considered rather limited and small compared to what is possible. In the remainder of this survey, we will see some clever protocols [Clark and Essex 2012, Maxwell 2013b] which make heavy use of it. Constantly new developed applications suggest that there is a large dormant, unexplored feature space.

As a playground to experiment with the Bitcoin protocol and its scripting capabilities, we recommend the Bitcoin testnet [Bitcoin wiki 2014, /Testnet]. It uses a separate, distinct block chain, and so-called *faucets* provide coins for

free. Apart from a few minor parameter alterations, the testnet runs the same code as Bitcoin peers but provides a safe harbor for testing.

Beyond Bitcoin, the so-called second generation of cryptocurrencies [Willett 2013], such as Mastercoin (MSC, `mastercoin.org`), Counterparty (XCP, `counterparty.io`), and Ethereum (Ether, `ethereum.org`) implement a new transaction syntax with a fully-fledged (Turing-complete) scripting language [Wood 2014, Buterin 2014]. They follow the idea of smart contracts [Szabo 1997] and colored coins [Rosenfeld 2012b], which are understood as digital assets. These assets can be used to realize sophisticated financial instruments such as stocks with automatic dividend payouts or to manage and trade physical properties such as cars.

Most of these next-generation coins work on top of Bitcoin's block chain and are therefore called *on-chain currencies*. Since they encode their transactions into Bitcoin's transactions, they lack the validation of transactions by miners, because Bitcoin miners do not "understand" the new transaction types. However, the new protocol layer can build upon Bitcoin's strong foundation and its security. Furthermore, it will increase Bitcoin's value from which both will profit. Sometimes the analogy used to describe the relation between next-generation currencies and Bitcoin is that of HTTP and TCP/IP: HTTP provides an versatile application-layer protocol to the more fundamental transport and network layer. Similarly, the scripting languages are compared to the relation between JavaScript and HTML.

# 3. SECURITY

In this section, we discuss security risks and security implications of the Bitcoin protocol and system design. Since Bitcoin is a digital currency with a notable market value, motives to exploit weaknesses for profit are ubiquitous. Beyond double spending, the attack vectors include key recovery and transaction malleability, for example. However, the most fundamental fear is the so-called 51% attack, during the discussion of which we will also have the opportunity to explore some fundamental system properties of Bitcoin.

## 3.1 Wallets and Cryptography

In order to use Bitcoin, the first thing a user needs is a wallet. The wallet holds a public key pair, which is the best approximation of the user's account. Thus, obviously, it is essential to take protective means to secure the wallet [Bitcoin wiki 2014, /Securing_your_wallet].

Common Bitcoin terminology distinguishes a variety of wallet types, such as software, hardware, paper, brain and online wallets. Software wallets are one of the most common ways to use Bitcoin. For a software wallet, a locally running Bitcoin instance is necessary. The Bitcoin developer team release a reference implementation of the Bitcoin protocol (`bitcoin.org`). It is a full client which processes the whole block chain. However, there are many alternative implementations such as Armory (`bitcoinarmory.com`) or Electrum (`electrum.org`), which offer additional features. Online wallets such as `blockchain.info` or Coinbase (`coinbase.com`) are another popular way to participate. They either manage the wallet centrally, or they follow a hybrid approach where the wallet is stored encrypted and most operations are performed on the client side in the browser. All software and online wallets are inherently prone to security problems because an attacker gaining access to a targeted machine can also obtain access to the user's wallet.

The term hardware wallet summarizes a class of approaches which follow the idea of using a separate device that usually operates offline. Since the device is not directly connected to a network, it becomes much harder for an attacker to gain access. For an example see [Bamert et al. 2014]. More advanced and secure ways (at least if done right) are paper and brain wallets. A paper wallet stores the keys holding coins offline as a physical document. This way they are comparable to cash money. A brain wallet takes it a step further and stores keys in the user's mind by memorizing a passphrase. The passphrase is turned into a private key, from which the public key and the Bitcoin address are derived with Bitcoin's standard hashing and key derivation algorithms. To avoid dictionary or brute-force attacks, the phrase must be sufficiently long and must have a very high level of entropy.

We already noted that multi-signature transactions can be used to increase the security of wallets. For example BitGo (`bitgo.com`) offers online wallets with 2-of-3 multi-signature transactions. In the same manner, threshold signatures can be used to impede theft and add two-factor security to wallets [Goldfeder et al. 2014]. The idea comes directly from secret sharing [Shamir 1979]: the secret, in this case the private key, is split into shares. Any subset equal to or greater than a predefined threshold is able to reconstruct the secret, but any subset that is smaller gains no information about the secret. The key property of threshold signatures is that the key is never revealed. Participants directly construct the signature. This way, very much like multi-signatures, threshold signatures can be used to require $m$-of-$n$ shares in order to sign a transaction. However, threshold signatures look like regular P2PKH transactions in the block chain, because they mutually construct a single transaction signature. They are, consequently, indistinguishable from the majority of transactions and thus conceal the details of access control. Multi-signature transactions, on the other hand, can be signed independently and do not require multiple rounds of interaction. [Goldfeder et al. 2014] discuss the advantages and disadvantages of both approaches in detail. They argue that especially corporate environments could profit from threshold signatures.

In order to secure transactions, the Bitcoin protocol makes heavy use of elliptic curve cryptography [Miller 1985, Koblitz 1987]. For transaction signatures, the elliptic curve digital signature algorithm (ECDSA) as standardized by NIST [Gallagher and Kerry 2013] is used, parametrized by the `secp256k1` curve defined by [SECG 2000]. For example, take the standard P2PKH transaction script. The user needs to provide her public key and her signature to prove ownership. To provide a signature, the user chooses a per-signature random value, which must be kept secret and must not be used for more than one transaction. Otherwise, the secret key can be computed from the signature. Even partially bit-wise equal random values suffice to derive the private key [Howgrave-Graham and Smart 2001]. Thus, it is essential for the security of ECDSA to use unpredictable and distinct per-signature randomness for every signature. [Bos et al. 2013] inspected the block chain for instances of repeated signature nonces for the same public key. They found that 158 public keys reused the signature nonce in

more than one signature, making it possible to derive private keys. For all these public keys, the remaining balance is negligibly small (smaller than the transaction fees needed to transfer them). However, one single address claimed coins from 10 of the vulnerable accounts (in sum over 59 BTC) from March to October 2013. The coins appear to have been stolen by this address. [Bos et al. 2013] traced the incident back and found that one cause was the incorrectly seeded random number generator of `blockchain.info`'s JavaScript client.

Thefts due to hacked systems, buggy software or incorrect usage are probably the greatest security risk in Bitcoin. As we will see, there are more examples which confirm this view. However, the most prominent type of attack is double spending which we already touched in the previous section, and which we will now re-consider in more detail.

## 3.2 Double Spending

In an online scenario with a centralized bank and with coins that are distinguishable (e. .g., by serial numbers), double spending as discussed before is trivially detectable. However, early digital currencies often also considered offline scenarios [Law et al. 1996], which made it impossible to contact the bank to authorize the transaction. This made double spending a major issue, even if a central authority existed. Generally, the Bitcoin setting is an online scenario (though offline transactions have been considered, too [Dmitrienko et al. 2014]). However, Bitcoin doesn't have a bank—and the distributed ledger opens up other possibilities for double spending.

The two generally conceivable ways to deal with double spending are (i) detecting it after the fraud actually happened and identifying the adversary for prosecution, or (ii) trying to prevent it. The above mentioned early digital currency approaches followed the first path: they accepted the possibility of double spends and required randomized parts of identifiers in the transactions. In case of double spending, the bank could assemble these parts afterwards to identify the adversary. The first approaches to mitigate double spends used the help of third parties in witness or quorum mechanisms [Szabo 2005, Osipkov et al. 2007, Hoepman 2007]. These approaches are similar to Bitcoin's approach, but still come with the major flaw of being vulnerable to Sybil attacks.

Bitcoin protects against double spending through the rule that only previously unspent transaction outputs may be used in the input of a follow-up transaction, where this rule is enforced during transaction propagation and mining, and where the order of transactions is determined by their order in the block chain. This boils down to a distributed timestamping [Haber and Stornetta 1991] and consensus algorithm [Malkhi and Reiter 1998, Szabo 1998], which in turn can be understood as the Byzantine Generals problem [Lamport et al. 1982, Szabo 2003]. To protect from Sybil attacks [Douceur 2002], Bitcoin couples this to a random oracle, i. e., the proof of work. Thus, the capabilities of an adversary are limited by his computational power.

The authors of [Miller and LaViola Jr 2014] modeled the Bitcoin protocol along the lines of a Byzantine consensus algorithm. They showed that it indeed reaches consensus. In particular, under the assumption of synchronous communication Bitcoin achieves optimal Byzantine resilience, so-called $2f + 1$ *resilience*, even in the presence of adversaries. This means the system is safe as long as the honest nodes $n$ prevail the adversaries $f$ by the ratio of $n > 2f + 1$. In case of Bitcoin, the network is resilient to adversaries controlling less than half of the computational power. In the next paragraphs we will assess this property in detail and consider how such an attack manifests in the block chain.

Recalling what has been said before on block chain forks and re-gaining consensus, an attacker can exploit this mechanism to mount a double spend attack along the lines of the example given in the previous section. A more generic blueprint for such a double spend includes the following steps: (i) broadcast a regular transaction (e. g., paying for a product), (ii) secretly mine on a fork which builds on the last block and includes a conflicting transaction which uses the same outputs as in step (1), but pays the attacker instead of the seller, (iii) wait until the seller is confident (i. e., receives enough confirmations) and hands the product over, (iv) as soon as the secret fork is longer than the public chain, broadcast the respective blocks. Because the secret branch is longer, the network will consider it as the valid main block chain. The regular transaction becomes invalid and cannot (even when broadcast by the seller) be included in a block anymore. Accordingly, [Courtois 2014] criticizes the longest chain rule and notes that it actually tries to solve two distinct problems, namely which blocks and which transactions should be accepted. He suggest to employ separate rules.

In order to assess the success conditions for this type of double spend attacks, let us take a look how to model the race between the benign and the fraudulent block chain. As in [Nakamoto 2008a, Rosenfeld 2012a], we describe it as a binomial random walk. Assume that the hash rates and therefore the difficulty remain constant. Further assume the probability that an honest node finds the next block is $p$ and the probability that an attacker finds the next block is $q = 1 - p$. We denote the difference in heights between the fraudulent and the benign block chain by $z$. Whenever a block is found, $z$ changes by either $+1$ for a benign block or by $-1$ for a fraudulent block. Thus $z$ is given by

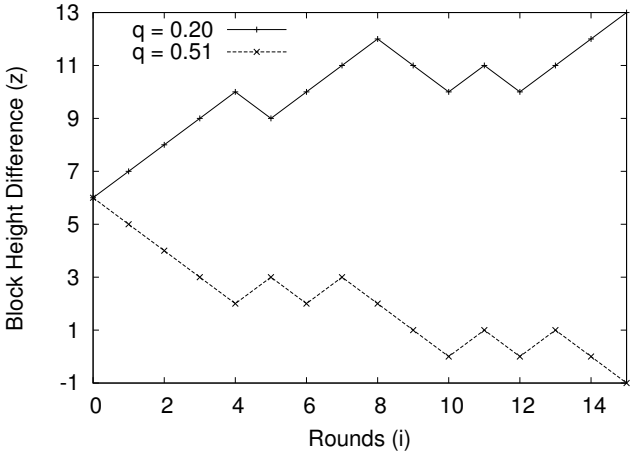
$$z_{i+1} = \begin{cases} z_i + 1 & \text{with probability } p \\ z_i - 1 & \text{with probability } q. \end{cases} \tag{2}$$

We are interested in the question whether $z$ will ever become $-1$, which implies that the attacker surpassed the benign block chain and successfully mounted a double spend. Figure 4a exemplarily shows the two possible outcomes in a random walk simulation. We can draw the first conclusion: if $q > p$, i. e., if the attacker controls more than the half of the total hash rate, he will succeed in catching up (for $i \to \infty$). The attacker's success is independent of the number of confirmations. This particular double spending attack is called the $> 50\%$ (sometimes also 51%) attack.

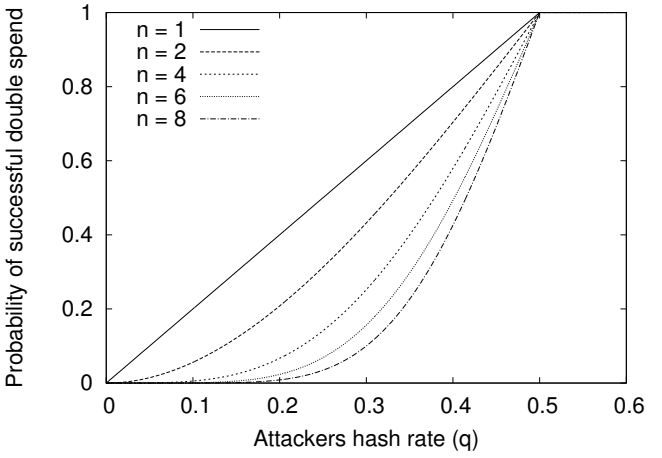
The situation is comparable to the Gambler's Ruin Problem [Feller 1957], which considers the probability of a player ending up without money in a coin-flipping game given the initial credit. For arbitrary probabilities $p$ and $q$ of the possible outcomes in a single iteration, it can be shown that the probability $q_z$ of experiencing gambler's ruin having started with $z$ credits yields

$$q_z = \begin{cases} 1 & \text{if } z < 0 \text{ or } q > p \\ (q/p)^z & \text{if } z \geq 0 \text{ and } q \leq p. \end{cases} \tag{3}$$

In case of Bitcoin, $q_z$ is the probability of the attacker ever




(a) Random walk simulation ($z_0 = 6$).




(b) Probability of successful double spending.

Figure 4: Hash rate-based double spending analysis.

catching up from $z$ blocks behind the benign block chain. While for $z < 0$ (the attacker already is ahead of the benign block chain) and $q > p$ (the attacker controls the majority of the overall hash rate) the attacker's success is certain, for the case $z \geq 0$ and $q \leq p$ the probability of success decreases exponentially with $z$.

Now, what is the probability of a successful double spend *while* the honest miners are finding new confirmations? Assume that the attacker has to wait for $n$ confirmations before the transaction is accepted by the victim. I. e., the attacker has to willingly "fall back" by $n$ blocks first in the current consensus chain. Meanwhile, though, the attacker is able to produce (not yet published) blocks in the fraudulent fork, the number of which we denote by $m$. The original Bitcoin paper [Nakamoto 2008a] assumes that $m$ follows a Poisson distribution. More accurately, [Rosenfeld 2012a] models the probability of $m$ as a negative binomial variable $P(m)$. Furthermore he assumes that the attacker pre-mined a block before initiating the attack, hence $z = n - m - 1$. It follows that the probability of a successful double spend equals

$$\begin{aligned} r &= \sum_{m=0}^{\infty} P(m) \cdot q_z = (...) \\ &= \begin{cases} 1 & \text{if } q \geq p \\ 1 - \sum_{m=0}^{n} \binom{m+n-1}{m} (p^n q^m - p^m q^n) & \text{if } q < p. \end{cases} \end{aligned} \tag{4}$$

We visualized the results of equation (4) for various numbers of confirmations $n$ in Figure 4b. Clearly, the higher the number of confirmations, the lower the probability of a successful double spend. The success probability converges to one when the attacker's hash rate approaches the 50% threshold. As [Rosenfeld 2012a] concludes from the results, double spending is possible with *any* hash rate of the attacker, and there is *no* number of confirmations that can reduce the success probability to zero: an attacker with less than 50% of the total computational power is able to perform a double spend by brute force and a bit of luck. A 51% attack will definitely lead to success. In this case, the attacker is able to claim all block rewards for himself, reject transactions and include only those he likes. This will likely drive miners off, which in turn increases the attacker's share and strengthens his position. Thus, the 51% attack is considered the worst-case scenario, because it will probably destroy the Bitcoin network. For this reason it is also called the Goldfinger attack [Kroll et al. 2013].

Traders accepting transactions immediately without any confirmation are particularly exposed to double spending. The authors of [Karame et al. 2012] analyzed double spending attacks for fast payments and demonstrated the feasibility. According to these results, it is possible for clients who are not miners to cheat. In order to mount a zero-confirmation attack, the attacker sends a transaction directly to the seller and broadcasts a double-spend transaction in another corner of the network at the same time. If the attacker is lucky enough, the double spend transaction will make it into the main block chain which is recognized by the seller too late.

By secretly pre-mining a block, the attacker can increase the chances of success. In every block, the attacker includes a self payment (this will become the double spend transaction). If he solves the proof of work, he suspends broadcasting and initiates a trade referring to the same coins. The seller (and even the network) will consider the transaction valid. As soon as the trade is completed, the attacker broadcasts the prepared block, which includes the double spend transaction and thus takes precedence over the other one. This attack was first described by Finney [Finney 2011] and is hence known as the Finney attack. The general strategy is also referred to as *block withholding attack*, which also finds use beyond double spending [Rosenfeld 2011, Courtois and Bahack 2014].

The fixes to zero-confirmation attacks are implemented since the beginning of Bitcoin: "just wait for confirmations". However, even when waiting for a confirmation, an attacker can employ strategies to increase the chances of success. The Vector76 attack [Vector67 2011] is an example of a 1-confirmation attack. Again, the attacker pre-mines and withholds a block, but this time includes a deposit transaction to the target (e. g., an exchange service) in this block. If the block is ready, the attacker waits for a block announce-

ment and quickly sends the pre-mined block directly to the target. The target and probably some miners will consider the pre-mined block as the main chain. Thus, the deposit transaction has one confirmation. In response, the attacker requests a withdrawal. If the attacker's fork of the block chain survives, the withdrawal will settle. If the other fork survives, however, the deposit is invalidated. If additionally the withdrawal does not use the same coins as the deposit (i. e., it does not refer to the outputs from the deposit), then it will still be valid and now in the possession of the attacker. The required behavior is not uncommon for exchanges.

The fix—again—consists of waiting for more confirmations. However, sellers (or exchanges) can also take some other precautions, like not accepting inbound connections, which we will detail when talking about the Bitcoin peer-to-peer network and its relay patterns in Section 4. For now, we conclude that race attacks for double spending cannot be eliminated entirely. Therefore, each participant of the Bitcoin network needs to trade off the risk and choose to wait for an appropriate number of confirmations. On the other side, there is a trade-off for the attacker, too, who also needs to consider the costs and benefits: if the attack fails, block rewards for pre-mined blocks will be lost. To compensate for the risk, it is therefore necessary to double spend a certain minimum amount.

Even when successful, the block chain allows to recognize double spendings and to identify the tainted coins [Gervais et al. 2014]. The victim will likely keep an eye on these coins and track their flow. Other traders might not be willing to accept tainted coins, because they will always be associated with a fraud. This led to blacklisting and whitelisting considerations. [Möser et al. 2014] provided first thoughts on quantifying and predicting the risks that are involved.

## 3.3 Transaction Malleability

Originally, transaction malleability refers to a bug in the Bitcoin protocol, which makes it possible to change the TXID without invalidating the transaction. Recall the anatomy of a transaction: it includes references to previous transactions (inputs) with a respective redeem script (`scriptSig`) and specifies one or multiple destinations (outputs). Each transaction can be uniquely identified by its TXID, which is a hash of the transaction data, including the redeem script(s).

The signature in a script, however, does not cover the same data as the hash which forms the TXID. In particular, it does not cover the redeem script. The rationale behind this design is that signing the script would imply signing the signature itself, which is obviously impossible. Instead, during signature generation, Bitcoin replaces all redeem scripts by a dummy script consisting of a single `OP_0` (this operation pushes an empty string on the stack). This substitution is not applied when calculating the TXID, though. As a consequence, it becomes possible to modify the transaction by substituting the redeem scipt by another valid (yet different) one. As a result, the signature remains valid, but the TXID changes.

For instance, a redeem script can be changed without invalidating it by pushing additional data to the stack prior to the data expected by the script of the connected output. In particular, as [Decker and Wattenhofer 2014] revealed, it was common practice for exploits to substitute the push operations by zero-padded alternative push operations, thereby preserving the semantics, but altering the TXID. But also the signature itself is vulnerable, because OpenSSL tolerates variations in the encoding of signatures. A list of known sources of malleability is provided by [Wuille 2014]. He also proposes some extra transaction validity rules to tackle most of the issues.

Transaction malleability is, of course, not desired. However, per se it does not cause any damage. An attacker can exploit the behavior, though, and make someone believe a transaction has failed, even though it later on gets confirmed. This becomes particularly relevant when targeting exchanges. Exchanges let users buy and sell coins for fiat money or altcoins and hence hold a significant amount of coins. They act as wallet providers to the effect that users who wish to trade need to provide a deposit. Exchanges are a point of failure in an otherwise highly distributed environment. Therefore they often employ cloud providers to protect them from distributed denial of service (DDoS) attacks [Vasek et al. 2014].

A transaction malleability attack against an exchange proceeds as follows: (i) the attacker withdraws coins from an exchange and (ii) as soon as the attacker receives the respective withdrawing transaction issued by the exchange, he rebroadcasts the altered version of this transaction with a different TXID. One of the two transactions eventually makes it into the block chain. Due to propagation delays and precautions the attacker can take, there is a chance that the modified transaction wins over the original withdrawal. If the exchange relies on TXIDs only, it will not find the withdrawal transaction in the block chain and believe the withdrawal has failed. As consequence, the attacker may withdraw again (and again).

Considered from this angle, the transaction malleability attack can be thought of as a variant of a double spending [Decker and Wattenhofer 2014]. In contrast to a typical double spend, however, the attacker is the receiving and not the spending party. The success of the attack depends on a number of constraints, i. e., the malleable transaction must be confirmed and the exchange must check for TXIDs only. [Andrychowicz et al. 2013] give further advice on how to issue malleability-resistant transactions without modifying the protocol.

Double spending attempts (including transaction malleability exploits) in general can only be observed while the colliding transactions are in circulation. Afterwards, only by parsing the block chain, it is not possible to identify successful instances. How exchanges implement the withdrawal process and whether they are truly vulnerable often remains unclear. Therefore, [Decker and Wattenhofer 2014] define a transaction malleability attack as successful if the altered transaction gets confirmed. The authors observed transaction activities on the Bitcoin network since January 2013 and identified 28,595 incidents of which approximately 20% were successful. They add up to 64,564 BTC which potentially got stolen.

Bitcoin's reference implementation is (and was) not affected, because it also tracks the respective UTXOs and takes them as an indication for a successfully issued transactions. Exchanges such as Mt. Gox—a popular exchange in the early days of Bitcoin—were apparently vulnerable, though: Mt. Gox released a statement that they were affected by transaction malleability attacks and that it is the cause for halting withdrawals and freezing accounts. [Decker

and Wattenhofer 2014] found a strong correlation between the press releases and the attack rate. However, if Mt. Gox stopped withdrawals as stated, it cannot be the root cause of their shutdown, because the majority of attacks occurred after the announcements: only 421 transactions adding up to approximately 1,800 BTC were potentially stolen before. The results rather suggest that the press releases motivated imitators to exploit the vulnerability elsewhere.

Due to the huge coin volume, exchanges are exposed to heist either by external attackers—or the operator. Based on public records including the block chain of 40 Bitcoin exchanges, [Moore and Christin 2013] model the risk coin holders face from exchange failures. They find that the transaction volume is an indicator whether or not the exchange is likely to close. They also confirm what might be intuitively obvious: less popular exchanges are more likely to close than popular ones, but popular exchanges are more likely to suffer from security breaches.

## 3.4 Pooled Mining

As pointed out before, Bitcoin must strive to maintain a structure in which no entity controls more than half of the computational power. Most of the time in Bitcoin's history, this was the case. However, being the first to successfully verify a block (i. e., being the first to find a valid nonce) happens only with a very small probability. Therefore, the payoff for *solo mining* is extremely bursty: a significant reward—but only very seldom. Miners therefore more and more group into *mining pools*. In a mining pool, multiple miners contribute to the block generation conjointly. Each participant searches parts of the nonce space for a valid nonce. In case one of them is successful, the profit is shared. Therefore, each participant get continuous small rewards, instead of seldom, large ones. Multiple different payout functions are used for sharing the profits in mining pools [Rosenfeld 2011].

From the security perspective, the trend to form mining pools raised concerns: a mining pool centrally aggregates computational power of miners who should, according to the key design idea behind Bitcoin, guarantee a valid distributed quorum through independent participation. A look at the block chain reveals that regularly multiple consecutive blocks are mined by a single mining pool. It already happened a few times that big mining pools such as `GHash.io` approached the critical threshold of 51% of the network's hash rate. And even if a single mining pool does not exceed the critical threshold by itself, coalitions are able to do so.

There are several options to tackle the problem. Actually, it is in the own interest of each miner to keep the distributed ecosystem intact. Therefore, the easiest solution is that miners by themselves switch to other mining pools so as to reasonably redistribute the power. As it turned out in the past, this works surprisingly well: calls by the community to switch mining pools were heard and the pools themselves started to support this movement [Hajdarbegovic 2014].

An engineering approach to the issue is to decentralize the mining pool. Even if a decentralized mining pool gains more than half of the computational power, this power cannot be exploited due to the lack of a central coordinating entity. P2Pool (`p2pool.in`) is a decentralized mining pool which builds a peer-to-peer network of miners. It creates a new block chain, called share chain. Indeed, the blocks of the share chain are valid Bitcoin blocks, but with a lower difficulty, so that every 30 seconds a new block is generated. If a P2Pool peer finds such a block, it gets broadcast, verified by others, and added to the share chain in similar manner as in Bitcoin itself. This continues until a peer finds a block that also meets Bitcoin's mining difficulty. The respective block is broadcast in the Bitcoin network and the reward is distributed among the P2Pool clients according to the share chain. Since the share chain blocks are required to include a coinbase transaction, which reflects the shares (i. e. the previously found blocks) and pays the miners accordingly, a successful miner cannot claim the reward for herself alone. P2Pool has some weaknesses, though, such as additional complexity and significant resource consumption, which might be the reason why it is not as attractive as its centralized counterparts.

Miners (and mining pools) compete with each other: their chance of winning is proportional to their computational power. However, rogue miners can achieve an unfairly high share by attacking the mining process. The typical intent behind these attacks is to weaken competitors with the aim to gain higher revenue. For that reason, mining pools are the second-most popular target of distributed denial of service attacks in the Bitcoin ecosystem [Vasek et al. 2014]. Actually, the trade-off between attacking or investing to gain an advantage can be expressed by rational game-theoretic models [Johnson et al. 2014, Laszka et al. 2015].

But miners can also exploit the mining process itself to gain an advantage: Instead of directly announcing mined blocks, miners keep their discovery private and establish a private chain. If the public chain approaches the length of the private chain, the rogue miner broadcasts his chain to catch up. This way miners intentionally force a block chain fork and initiate a block race. The key idea is to let honest miners waste their power by mining on the public chain, so as to increase the own chances of winning on subsequent blocks. The strategy is widely known as *selfish mining* [Eyal and Sirer 2014]. It can also be used to gain an advantage while participating in a mining pool [Rosenfeld 2011, Eyal 2014]. More generally, the attack vector is called *block withholding attack* [Courtois and Bahack 2014] or *block discarding attack* [Bahack 2013]. For example, the previously mentioned Finney attack also falls into this category (but with the aim of double spending). The problem of selfish mining is that it increases transaction approval time and facilitates double spending.

[Eyal and Sirer 2014] formally analyzed the incentives for selfish mining and showed that selfish miners obtain a revenue larger than their relative share. The success and profitability heavily depends on the selfish miner's share $q$ of hash rate and the fraction of honest miners that choose to mine on the private chain after broadcasting. The latter is denoted by $\gamma$ and often depends on network properties (i. e., who hears which block first). In particular, [Eyal and Sirer 2014] made the observation that selfish mining is profitable if

$$\frac{1-\gamma}{3-2\gamma} < q < \frac{1}{2} \quad . \tag{5}$$

Since selfish (or adversarial) miners who control more than half of the hash rate asymptotically always win the race and are able to catch up with public chains, we are interested in the case $q < 0.5$ only. For $\gamma \approx 1$, which implies that (almost) all honest miners favor the private over the public chain, selfish mining is profitable for virtually all shares $q$. In the

other extreme case of $\gamma \approx 0$, where (almost) all honest miners disregard the private chain, the profitability threshold of selfish mining equals a share of at least 1/3. That is, miners with more than one third of the computational power can increase their revenue by following the selfish mining strategy. Note that, depending on $\gamma$, the threshold ranges between 0 and 1/3 and thus is substantially lower than the usually considered critical hashing power of $q > 0.5$. Solo miners are unlikely to deliver such a performance nowadays, but mining pools are very well able to do so. The lower bound given by the left hand side of (5) can be considered a security metric: the higher this security threshold, the higher the fraction of the total computational power that is necessary to exploit selfish mining.

The true, current value of $\gamma$ (and thus of the security threshold) is unknown. By mounting eclipse attacks, which we will discuss soon, an adversary is likely able to push $\gamma$ closer to one and thus to lower the security threshold significantly. Therefore, [Eyal and Sirer 2014] propose to relay all blocks that are received within a certain timespan and to select the fork to mine on randomly. As a result, half of the honest miners will choose the private block, which fixes $\gamma$ at 0.5. This yields a security threshold of 0.25. [Heilman 2014] raises the threshold to 0.32 which renders selfish mining ineffective. Their mitigation is called "freshness preferred"; it suggests to choose the most recent block (according to its timestamp). Yet, it assumes unforgeable and accurate timestamps, which is not easy to achieve [corbixgwelt 2011, Cohen 2014].

Bitcoin peers maintain a network time, which is the median of time samples from their neighbors. It is secured from arbitrary manipulation by allowing at most a deviation of 70 minutes from the system time. For the sake of triple modular redundancy ("never go to sea with two chronometers; take one or three."), the user is asked to double check the time. Nevertheless, an adversary is able to slow down or to speed up nodes within a tolerance range of 70 minutes. This attack is known as *timejacking* [corbixgwelt 2011]. An advanced attacker can use timejacking to isolate a miner. By speeding up the majority of clocks while slowing down the target's clock, the attacker can achieve a difference of 140 minutes. Since the network time is used to validate blocks, the attacker can generate a "poison pill" block with a custom timestamp, which is accepted by the majority but rejected by the target. As a result, the target sticks to the previous chain and continues mining on this part, whereas the majority of the network has moved on. All newly generated blocks are immediately rejected by the target. Timejacking can be considered a form of denial of service attack, but it also facilitates double spending. Furthermore, it can be used to influence the mining difficulty calculation [Cohen 2014]. The proposed solutions include to tighten the tolerance ranges, to use NTP, or to use trusted peers for time sampling only [corbixgwelt 2011].

Most of the attacks discussed in this section are possible because of Bitcoin's network structure. As we will see next, significant propagation delays and scalability issues of the network are often the root causes.

# 4. NETWORK

In this section, we will take a closer look on how Bitcoin organizes the distributed network of peers. In particular, we will consider the Bitcoin protocol, the resulting relay patterns and their implications on information propagation. Furthermore, we provide an outlook on Bitcoin-inspired network applications and services, such as alternative domain name and messaging systems.

Bitcoin uses an unstructured peer-to-peer network based on persistent TCP connections as its foundational communication structure. In general, unstructured overlays are easily constructed and robust against high churn (change-and-turn) rates, i. e., against frequently joining and leaving peers. From existing research, it is known that unstructured overlays like, for instance, Gnutella [Gnutella v0.4 2003] do not scale well [Lv et al. 2002, Chawathe et al. 2003]. Searching for files in unstructured file sharing overlays, for instance, requires flooding requests in the network, so that each peer receiving and forwarding a query can check against the locally known data items. This causes significant overhead due to the massive number of copies of each query, and because of the need to maintain state information about seen messages for duplicate suppression. The load on each peer grows linearly with the system size. In order to reduce the amount of relaying, the scope of queries is often limited to a certain number of hops in peer-to-peer file sharing systems—at the cost of imperfect coverage: not every query reaches every peer.

The Bitcoin network has aims which differ from those of peer-to-peer file sharing systems. In Bitcoin, the aim is not to find specific files or data items, but to distribute information as fast as possible to reach consensus on the block chain. Limiting the scope of message propagation is therefore not an option. Clearly, this raises concerns regarding Bitcoin's ability to scale to higher transaction rates while still processing transactions rapidly.

## 4.1 Joining and Maintaining the Network

Every peer in the Bitcoin network actively tries to maintain a minimum of eight connections in the overlay. That is, the peer tries to establish additional connections if this number is underrun. The number of eight connections can be significantly exceeded if incoming connections are accepted by a Bitcoin peer; usually a network participant does not handle more than 125 connections at a time (`maxconnections`). By default, peers listen on port 8333 for inbound connections. When peers establish a new connection, they perform an application layer handshake, consisting of `version` and `verack` messages. The messages include a timestamp for time synchronization, IP addresses, and the protocol version. Since Bitcoin version 0.7, IPv6 is supported.

In order to detect when peers have left, Bitcoin uses a soft-state approach. If 30 minutes have been passed since messages were last exchanged between neighbors, peers will transmit a heartbeat message to keep the connection alive. If 90 minutes have passed without any incoming message, the client will assume that its counterpart is offline. Bitcoin peers also keep track of not directly connected peers in the network. They maintain a list of recently active peers, including their IP address and a timestamp. Every peer broadcasts its own IP address in an `addr` message every 24 hours through the overlay. The absence of the message from a certain peer is interpreted as a sign that the respective peer is now offline. Exchanging `addr` messages (during bootstrapping and later on) is the common way to explore the network.

Besides unsolicited reception of `addr` messages, peers can ask neighbors for additional peers by sending a `getaddr` message. The response (`addr`) contains a random selection of 23% (but not more than 1,000) peers from the responder's list of recently active peers. (It seems that there are no particular reasons for the choice of these numbers.) Note that this list does not only include peers directly connected to the originator of the list, but also peers it heard of recently. Thus, retrieving `addr` messages does not reveal the structure of the network without additional effort. In fact, it is a design goal of the Bitcoin network implementation to obfuscate the topology and to make sure that (local) attackers cannot fill up a peer's neighbor table with compromised IP addresses. Otherwise it would facilitate eclipse attacks (in the context of Bitcoin also often called a netsplit), where an attacker monopolizes or at least dominates the environment of a node, therefore controlling the message flow between this node and the remainder of the network. An attacker can exploit this to generate an independent, inconsistent view of the network (and the block chain) at the attacked node. This enables double spends with more than one confirmation. Nevertheless, since peers get regular updates and maintain all received participant information, every peer has a relatively broad overview of the peers in the network.

Whenever receiving an `addr` message, peers consider relaying address information for messages with a maximum of ten addresses only. If the timestamp associated with the information about a peer is not older than 10 minutes, the peer decides to relay the respective address. Depending on the reachability of a peer, i. e., whether the advertised IP lies within a reachable network, the address is either forwarded to two neighbors or one. This promotes peers with publicly IP addresses more than peers with private addresses behind a NAT, based on the conjecture that peers with public IPs are more likely to accept incoming connections. The decision which neighbors will receive the address information is determined by deterministic randomness (i. e., a pseudorandom decision based on a fixed seed). Peers maintain state of already advertised IP addresses to avoid repeated advertisements. The seed changes every 24 hours. Addresses designated for the same neighbor are collected in a new `addr` message and relayed in a batch. About every 100 ms, a neighbor is randomly selected (the so-called "trickle node") and the queued addresses are flushed. This mechanism induces an additional random delay per hop during address propagation.

Bitcoin peers use three methods of finding neighbors during bootstrapping: DNS, IRC, and asking neighbors. Since Bitcoin version 0.6, DNS is the default bootstrapping mechanism. The software is shipped with built-in hostnames of seed nodes, the IP addresses of which are resolved via DNS. The DNS servers are run by volunteers and return a set of recently active peers. The usage of IRC, where IP addresses are encoded in the nicknames, is replaced by DNS bootstrapping. Besides, the client asks its neighbors for a list of available peers as introduced above.

In a study from November 2013 to January 2014, the authors of [Donet et al. 2014] asked a set of initial peers for information about other peers they know (i. e., by sending a `getaddr` to these peers). For every previously unknown peer thus discovered, they repeated the procedure and asked them for peers too. In the first round they already discovered 111,475 IP addresses. After 37 rounds during the 37 days of the study, they discovered 872,648 IP addresses in sum. Geolocation lookups revealed that they are spread all over the world. Most of these peers are located in the US (22%) and China (14%). However, considering the number of Internet users in the respective countries, the Netherlands and Norway show the highest adoption rates. The discovered peers in each round show a significant overlap. However, most of them were gone after five days and only 5,769 were online throughout the whole period. Note that some of the peers have dynamic IP addresses and therefore may appear to be more unstable than they really are. [Feld et al. 2014] provides statistics on the AS level. The online service Bitnodes (`getaddr.bitnodes.io`) maps the global Bitcoin node distribution on a regular basis. All reveal the dimension of the Bitcoin network.

## 4.2 Transaction and Block Propagation

Based on the unstructured overlay as discussed so far, information about new transactions and blocks is spread to the peers in order to form the distributed consensus. The mechanism is simple: messages are flooded through the network. Let us revisit our example where Alice wants to issue a transaction. Where we focused on the semantics and the transaction and block chain structure before, we now take a closer look on the exchanged messages. The message flow is illustrated in Figure 5. Assume Alice constructed a valid transaction. Before broadcasting the actual transaction data including all details of inputs and outputs, she sends an inventory (`inv`) message stating "I know about new transactions" to all of her neighbors. This message contains a list of transaction hashes (TXIDs), but not the actual transaction data.

Alice's neighbors will request data from specific transactions in a separate `getdata` message, if these transactions are so far unknown to them. This pull-based communication mechanism reduces the load on the network by avoiding unnecessary, redundant transmissions of transaction records. In response to a `getdata` message, Alice sends the respective transaction record. After Alice's neighbors verified the transaction, they will make it available to all their neighbors in the same manner as Alice did, starting with an `inv` message.

Messages in general are flushed periodically about every 100 ms. However, transaction relaying takes place by "trickling" messages out. Bitcoin randomly selects with a probability of 1/4 the transactions for an `inv` message and stalls the remaining transactions. Every neighbor gets a different set of randomly chosen transactions (about 1/4 of the currently available set). Only the randomly selected "trickling node" (cf. `addr` message relaying) gets all transactions immediately. The other neighbors either get it later or already got it from another neighbor. Trickling reduces the overhead and at the same time makes traffic analysis more difficult, in a similar manner as mixes do in mix networks [Chaum 1981]. It also helps to conceal the originator of a transaction. However, since every 100 ms a random batch is flushed, an additional delay in the order of some hundreds of milliseconds is induced to the propagation of transactions. Therefore, trickling in Bitcoin is, in fact, counterproductive to the aim of propagating information as fast as possible. It trades off overhead and privacy against fast transaction propagation.

Peers keep track of the transactions that they have already seen, but "forget" them after a while if they do not

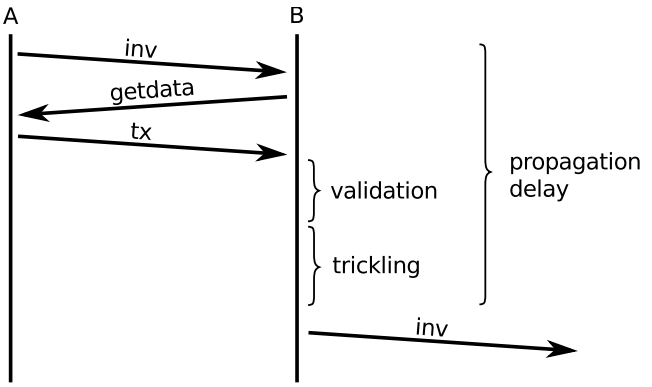



Figure 5: Transaction propagation.

make it into the block chain. Alice, as the originator of her transaction, is responsible for its distribution. She might hence need to re-broadcast it if the transaction did not get into the block chain, to make sure it gets considered in the next block.

By observing the forwarded messages from a highly connected peer, [Koshy et al. 2014] revealed three distinct relay patterns. With about 91% of all observed instances, the most common relay pattern involves lots of peers relaying a transaction only once. Since clients keep track of seen transactions and relay only new ones, this is exactly what one would expect. The second relay pattern involves a transaction received once or multiple times from a single peer. This is not very common (3%); it occurs when invalid transactions are broadcast and hence not relayed. The third relay pattern involves a transaction relayed by multiple peers and re-relayed by at least one of them (6%). The reason behind the occurrence of this pattern is that transaction originators are responsible for their transactions and might need to re-broadcast if they get forgotten.

The insights into transaction propagation also help to increase the resilience to fraudulent users paying with Bitcoin. For example, assume Alice wants to buy a snack and pay with Bitcoin. Further assume that Alice is very hungry and wants the snack asap, without waiting for the commonly expected six confirmations (about 1 hour). On the other hand, the merchant wants to make sure that it is safe to hand over the product without fearing the risk of double spending (we discussed the risk of double spending in such fast payment scenarios before [Karame et al. 2012]). With this storyline in mind, the authors of [Bamert et al. 2013] developed propagation strategies for merchants to realize fast payments. From the merchant's perspective, the key observation is that Alice should not be the only source of information for this transaction. Quite in contrast: the merchant should not accept incoming connections and connect to a preferably large sample of peers. In this case, Alice is forced to broadcast the transaction; she cannot directly send transactions to the merchant. In order to avoid isolation and timing attacks, the merchant should not relay transactions. As soon as he is connected to at least one honest peer, he will receive potentially fraudulent transactions and is able to recognize the double spending.

Another (not implemented) proposal is to use Bitcoin's `alert` messages, which are aggressively flooded, to signal a double spending attempt [Karame et al. 2012]. The intuition is that even when the attacker is able to fool the merchant, some (honest) peers will see the colliding transactions. They can then broadcast alerts immediately. Thus, by considering these strategies and waiting for a small listening period, the merchant makes sure that the majority of the Bitcoin network has received the transaction. Additionally he probably received Alice's transaction more than once from his neighbors. If there is no attempt to double spend, the risk of fraud has been reduced. Nevertheless, a cost-benefit trade-off remains.

The propagation of validated blocks is analogous to the propagation of transactions. A miner who has successfully solved the proof-of-work broadcasts an inventory message to all neighbors first. The full block is transferred upon request only. Peers receiving a new (unseen) block will relay it in the same manner. The period from receiving an inventory message for a new block until forwarding the announcement to all other neighbors induces a propagation delay, which consists of the time it takes to announce, request, transmit and validate the block (or transaction). In case of blocks, the validation includes the validation of each transaction and hence access to the block chain. In order to reduce the propagation delay, trickling is not used for blocks. Nevertheless, the induced propagation delay has implications on the Bitcoin protocol.

In [Decker and Wattenhofer 2013], the authors analyzed the information propagation in the Bitcoin network. They connected to a large sample of nodes in the network as observer, i. e., without actively relaying messages. They registered the arrival times of block hashes in `inv` messages from these nodes. The probability density function of times since the first block was received shows an exponential distribution, with a median of about of 6.5 seconds and a mean of 12.6 seconds. The distribution shows a long-tail behavior with 5% of the nodes still not having received the block after 40 seconds. The authors also discuss the relationship between the propagation delay and the probability of block chain forks. Obviously, due to the significant propagation delay, forks in the block chain become more likely. Thus, they conclude that if the amount of transactions and/or the network size increases, propagation delays will raise and, consequently, the rate of block chain forks will increase, too. This has an important impact on the resilience of Bitcoin against malicious nodes [Garay et al. 2014]. In order to mitigate the threat, [Decker and Wattenhofer 2013] propose a modified message exchange behavior. It aims to reduce the propagation delay by pipelining the block propagation, and by splitting the validation into checking for a valid nonce first and validating all transactions later on. However, this introduces new attack vectors, allowing adversaries (or everybody else) to flood `inv` messages through the whole network without providing a valid block or transaction.

In summary, the key determinant for information propagation in the Bitcoin network is the unstructured overlay network and its characteristics. As the network grows, its diameter will increase respectively. Thus, [Decker and Wattenhofer 2013] minimized the distance to mitigate the issues by deploying a highly connected peer (approximately 3,500 neighbors) in the live Bitcoin network. Their client actively tried to connect to every peer from the address list, reducing the distance between any two nodes to, ideally, two hops only. As a consequence, the block chain fork rate dropped from 1.69% to 0.78%—but at the cost of bandwidth require-

ments peaking at around 100 MB/s. This reveals a pressing issue of the Bitcoin network today: its scalability.

## 4.3 Scalability

The main objective of the peer-to-peer network in Bitcoin is to quickly distribute the information into every part of the network. Variations in the propagation mechanisms directly affect the formation of the distributed consensus and thus the security of Bitcoin. In general, inconsistent states, i. e., block chain forks, are undesirable, because they facilitate double spending. However, the Bitcoin network is faced with scalability issues. Especially network bandwidth, network size and storage requirements pose challenges.

Bitcoin's wiki states that the protocol is capable of much more than the current transaction rate [Bitcoin wiki 2014, /Scalability] and thus able to scale to higher demands. Currently, Bitcoin has an artificial maximum block size of 1 MiB, which limits the number of transactions per block and therefore also the growth rate of the block chain. This limit is enforced to prevent from ballooning the block chain before the protocol is prepared. The smallest standard transaction (which is not unspendable or spendable by anyone), is a single-input, single-output "pay-to-PubKey" (P2PK) transaction, which has a size of 166 bytes. A back-of-the-envelope calculation results in a theoretical upper bound of approximately 10 transactions per second (tps). A more conservative and realistic assumption would be to consider P2PKH transactions with at least two inputs (to merge previous outputs) and two outputs (for change). Accordingly, Bitcoin is capable of a transaction rate of approximately 4 tps. Alternatively, the block generation interval could be shortened, which implies that the proof-of-work difficulty would have to be adjusted accordingly. As discussed before, though, close-to-simultaneous block validations by different miners lead to block chain forks. Therefore, shorter block creation intervals come at the price of a higher chance of block chain forks.

Either way, scaling to higher transaction rates will eventually consume more resources. For example, to handle a rate of 2,000 tps, a block size of more than half a gigabyte and an Internet connection of approximately 1 MiB/s is required. As [Kaminsky 2011] states, a higher transaction rate (which is inevitable if Bitcoin really poses an alternative to the banking model) will eventually demand a super peer-based overlay structure (as in later versions of Gnutella [Klingberg and Manfredi 2002]) in order to handle the load. We can observe some evidence of this trend—and the emergence of super peers in the Bitcoin network. For instance, the study of [Donet et al. 2014] found that from 1,300 connected peers, 20 forwarded more than 70% of both transactions and blocks first. Therefore, [Reed 2014] suggests to explicitly introduce a hierarchical network structure, which consists of super peers (i. e., miners), full nodes (e. g., exchanges) and wallet nodes (e. g., online wallets or thin clients).

*Full nodes* (or *full chain clients*) download and verify all blocks starting from the genesis block. This is the most secure mode of operation. Even though not strictly necessary for a client, full nodes participate in the P2P network and help to propagate information. Alternatively, *thin clients* which use the simplified payment verification (SPV) [Nakamoto 2008a] can be used. A thin client needs the block headers only and requests transactions on demand. The block header incorporates a Merkle root [Merkle 1987], which secures the transactions of that particular block by constructing a hash tree over the TXIDs: transaction hashes are paired and hashed. Hashes without a partner are hashed by themselves. This is hierarchically repeated until a single hash remains, the Merkle root. It allows clients to verify that a transaction is part of a block by starting at the respective leaf and traversing the branches up to the root. If the final hash equals the Merkle root, the transactions must be part of the block. For the verification, thin clients request a list of (some) intermediary hashes from full nodes, but they do not need the complete block with all transaction data. Since the block headers including the Merkle root are secured, valid intermediate hashes cannot be faked easily. Thus, the approach can reliably verify the existence of transactions. Their absence, however, can be faked by answering with invalid hashes.

Clients can reduce the risk by sampling from multiple nodes. Yet, eclipse attacks are of course possible. In addition to security issues, requesting specific transactions from full nodes has important privacy implications: from the requested transaction, a full node might infer the owner of the coins. To limit the information leak, Bitcoin employs a Bloom filter [Bloom 1970] (a probabilistic data structure) to obfuscate requests [Hearn and Corallo 2012]. Clients express their request as a Bloom filter and send it to the full node. All transactions which match the pattern given by the Bloom filter are sent to the thin client. The inherent false positive rate of Bloom filters is used to adjust the desired level of privacy at the expense of some additional overhead.

Thin clients mitigate some of Bitcoin's scalability issues by relying on the data provisioned by full nodes. Even though they employ techniques to limit the necessary trust in others, thin clients subvert Bitcoin's core intent of rigorous decentralization. In a sense, the resulting structure resembles the banking model [Kaminsky 2011]. Thus, let us take look at other approaches which tackle the root cause of Bitcoin's scalability issues.

A fundamental bottleneck is the sheer size of the block chain. Since Bitcoin version 0.8, transactions and block indices are stored with LevelDB instead of the previously used Berkeley DB. This improvement increased the performance of synchronization and block verification, which used to be a bottleneck before. Nevertheless, the storage issues remain.

When considering thin clients anyway, a consequent step is to separately store the raw transaction data, and to include transaction hashes in blocks only. Indeed some altcoins such as Dogecoin (DOGE, `dogecoin.com`) follow this approach. The approach still required the distinction between thin clients and full nodes. In particular, full nodes still need to store all data. The author of [Bruce 2014] takes it one step further and disassembles the block chain into its components. He isolates three key components. First, the block chain manages ownership records and thus implicitly account balances. Second, it helps the network coordinating transactions. Third, the linked blocks and the proof of work secure the ledger. For each function, he suggests a data structure which takes the responsibility of the respective function, with the overall aim of substituting and slimming down the block chain. The account balances are tracked in a so-called account tree. It combines a binary radix tree and Merkle hashing with UTXOs as leafs. The roots of the radix tree and the Merkle tree become part of

the block header. This ensures integrity of the ownerships and supports quick address lookups. In order to periodically group transactions and update the ledger, a component analogous to the block chain is necessary. However, due to the account tree it becomes possible to discard old blocks. Thus, the proposed solution is to keep a few hundreds of blocks in a so-called mini block chain only. Therefore, inputs and outputs of transactions do not point to other transactions anymore. Instead, they point to addresses in the account tree and are thus implicitly linked. Simply discarding old blocks weakens the security, though. The solution is similar to the thin clients' strategy: a proof chain with block headers only. All three data structures together use less space but provide the same functionality. The design is not meant to substitute the live Bitcoin system: migrating the complete block chain seems impossible. An alternative currency, named Cryptonite (XCN, `cryptonite.info`), employs the proposed mini block chain scheme.

Even if we assume Bitcoin to be able to adapt to higher loads, there are additional limitations on the transaction rates [Sompolinsky and Zohar 2013]. Especially delayed block propagation and Bitcoin's security assumptions restrict transaction rates more than the limits imposed by, for example, bandwidth requirements. A fact noted by [Decker and Wattenhofer 2013] and [Sompolinsky and Zohar 2013] is that attempting to increase either the block creation time or the block size not only increases the bandwidth requirements, but also adversely affects the protocol. If the block that has been verified is not propagated rapidly, the probability of a fork increases. Bitcoin may be able to resolve forks, but frequent conflicts still waste valuable resources. Obviously, larger blocks imply longer propagation times and thus increase the risk of forks. [Sompolinsky and Zohar 2013] used the data set of [Decker and Wattenhofer 2013] and revealed a linear dependency between the block size and the propagation delays in the Bitcoin network. By extrapolating from the data set, they found that it takes 0.066 s/KiB to reach half of the nodes in the network.

Higher block generation rates will likewise result in more frequent conflicting blocks. The authors of [Sompolinsky and Zohar 2013] provide estimates of transaction rates as a function of the block size and the block propagation delay with regard to delays and security guarantees. Bitcoin's security heavily depends on the assumption that the time it takes to propagate blocks is significantly shorter than the block generation time. Thus, with an increasing transaction rate it becomes more and more likely that attacks are possible for an attacker controlling less than 50% of the overall hash rate. In an (optimistic) estimate, [Sompolinsky and Zohar 2013] assumed an unlimited block size, a transaction size of 0.5 KiB, and an adversary controlling 40% of the network's hash rate. Furthermore, they used the propagation delay factor of 0.066 s/KiB from above. The resulting upper bound on the transaction rate is approximately 40 tps.

As an optimization, [Sompolinsky and Zohar 2013] suggest to alter the mechanism to resolve block chain forks. The basic observation is that orphaned forks include valid blocks, which can contribute to the block chain's irreversibility. In particular, instead of the longest block chain, the fork with the heaviest subtree should be taken as a metric to resolve conflicts. It exploits the work that was already invested and enhances the security of their ancestor. It is thus possible to increase the transaction rate under the same security constraints.

We can conclude that the general scalability issues of unstructured overlays combined with the issues induced by the Bitcoin protocol itself remain. Some practical, pragmatic solutions appear able to keep Bitcoin in a working state for the foreseeable future. However, many of the results suggest that scalability remains an open problem. A summary of the most prevalent issues is provided by [Courtois et al. 2014a]. The considerations also raise the question whether Bitcoin is (and can remain) a decentralized currency [Laurie 2011, Gervais et al. 2014, Kroll et al. 2013].

## 4.4 Deanonymization

Tracking message flows does not only help to understand the network, it also discloses user information. [Kaminsky 2011] noted that by controlling a hub connected to all peers, it is possible to learn the IP address of any transaction originator: assuming that the transaction is not forwarded by an online wallet provider, the originator is likely also the issuer of the transaction. This breaks the pseudonymity of transactions. The relay patterns identified by [Koshy et al. 2014] confirm the expected behavior. Inspired by [Kaminsky 2011], the authors used their insights and developed heuristics to match transactions to IP addresses, even if the observing hub is not fully connected. During their study of five months, they were able to link 1,162 bitcoin addresses to IP addresses while being connected to a median of 2,678 peers.

Internet anonymity services like Tor [Dingledine et al. 2004] provide a solution to this privacy issue by concealing the originating IP. Tor decouples sender and receiver information by employing the onion routing protocol [Goldschlag et al. 1996] along a circuit of relays nodes. The IP address of the source node is thereby hidden, the destination of the connection sees only the address of the last Tor node along the circuit (the *exit node*). Therefore the Bitcoin client is able to tunnel the traffic through Tor via the SOCKS interface.

However, [Biryukov et al. 2014] point out that it is possible for an attacker to trigger a ban of Tor connections to the Bitcoin network. They exploit Bitcoin's denial-of-service protection, which blacklists misbehaving nodes under certain circumstances. Whenever a Bitcoin peer receives malformed messages, it increases a penalty score for the respective IP address. If the score hits a threshold, the IP is banned from connecting to this Bitcoin peer for 24 hours. One possibility for a simple, small message which is malformed in the sense of this mechanism is to send blocks with an empty transaction list. An attacker could use a Tor circuit for each pair of Tor exit node and Bitcoin peer, and mount a straightforward denial-of-service attack by getting the Bitcoin peers to blacklist all Tor exit IP addresses. In a similar way, other proxies can be banned from the Bitcoin network. Such an attack involves many connections and large amounts of traffic, but nevertheless seems feasible. Bitcoin over Tor in general and mounting the mentioned denial-of-service attack in particular introduces additional attack vectors, such as eclipse attacks by banning Tor from benign Bitcoin peers only or man-in-the-middle attacks by blacklisting benign exit nodes from Bitcoin peers [Biryukov and Pustogarov 2014a].

But even when using anonymized connections, it is possible to map the originator of transactions. [Biryukov et al. 2014] made the observation that a peer's set of neighbors can serve as a fingerprint. Recall that clients usually connect to

eight peers and advertise their addresses in the network by broadcasting it to all neighbors. The authors call the eight peers (which apparently accept incoming connections) the client's *entry nodes*. Clients maintain connections to entry nodes as long as they remain reachable. Thus, it can be assumed that the fingerprint is stable. If an adversary is already connected to the network, he will receive the address announcement, too. An adversary can exploit this fact and create a fingerprint for an IP address by connecting (ideally) to all Bitcoin servers and logging the set of peers that forward the IP address. These peers are likely the entry nodes for the respective IP. Due to the "trickling" of `addr` messages, an adversary will see a fraction of the entry nodes only. This and timing effects can result in false positives, though. In the next step, the attacker is able to map transactions to entry nodes and thus also to the originator of the transaction. If the transaction is relayed by a subset of the entry nodes, it can be linked to the respective client. In detail, this is tricky because of network latency and trickling, but as the authors show in their experimental results, the attack still has a significant success rate (about 11% of all transaction could be disclosed).

The proposed mitigation strategy from [Biryukov et al. 2014] suggests to rotate outbound connections, for instance after every transaction, so as to blur the fingerprint. This proposal started a discussion [Lists 2014b] that resembles many arguments also used in a very different context, namely the entry selection policy for the Tor anonymity network [Øverlier and Syverson 2006, Elahi et al. 2012, Dingledine et al. 2014]. The arguments include that rotating entry nodes periodically will lead to a higher probability of selecting a malicious entry node. Sticking to a stable set of entry nodes reduces this risk. Eventually, as for Tor, it will be necessary to differentiate with respect to assumed adversary capabilities, to clearly state the attacker model, and to trade off the implications of any defense.

## 4.5 Botnets

Botnets are a distributed formation of processes connected to a network. Illegal botnets run on systems without the knowledge of their operator. They have access to local files, network resources, and are able to run arbitrary programs. In most cases they communicate with a botmaster over a so-called command and control (C&C) channel. The botmaster uses it to seed new instructions to and to collect information from the bots. There are several approaches, such as IRC, Tor, or distributed hash tables (DHT), to realize the C&C channel. [Ali et al. 2015] shows how to use Bitcoin as a C&C infrastructure by encoding instructions in the transaction scripts. The approach benefits from Bitcoin's resilience.

Botnets are often used with the purpose of making money, including phishing and sending spam emails, but also distributed denial of service attacks. Thus, it was only a matter of time until botmasters would discover crypto currencies as an additional source of income. The authors of [Plohmann and Gerhards-Padilla 2012] perform a case study on an early adopter, the Miner Botnet (first activities date back to December 2010). Technically, the botnet is not state of the art, but at this early time it was distinguished by its mining capabilities. In particular, worker bots perform benchmarks on the compromised system and retrieve detailed information on graphic cards to initialize the mining software. The worker bots connect to proxy bots who collect the results of their work. The proxy bots run the standard Bitcoin client software and connect to a randomly selected mining pool from a hard-coded list. Every 20 minutes, proxy bots post their wallet holding the minted coins to the C&C server. The geographical distribution of the Miner botnet clusters around the countries Ukraine, Russia, Poland, Romania, and Belarus, which is likely due to the spreading strategy adopted by the Botnet owner.

The authors of [Huang et al. 2014] termed the above mentioned approach *proxied pool mining*. By investigating several other botnets, they identified additional strategies, which they called *direct pool mining* and *dark pool mining*. As the name implies, with direct pool mining, boneed to rets do not need a proxy and directly connect to a mining pool. The approach is very simple and does not require a separate proxy infrastructure, but it also has some disadvantages. Mining pools can easily detect botnet mining, because of the large number of miners with small hash rates, all sharing the same account. Proxied pool mining can be detected, too, but masks the worker bots and has the flexibility to quickly switch to a new proxy if banned. With dark pool mining, the botnet hosts its own mining pool to which workers connect. The mining pool server consequentially connects to the Bitcoin network. The earnings for direct and proxied pool mining flow constantly due to the public mining pools. Dark pool mining will result in bursts of earnings. [Huang et al. 2014] analyzed the profitability of mining botnets and concludes that throughout most of 2012 and the first quarter of 2013 it was absolutely profitable, even considering the prices charged (in dark corners of the Internet) for hiring a botnet. [Heusser 2013, Dev 2014] propose techniques to accelerate mining with non-custom hardware, i. e., CPUs and GPUs, of which Botnets can make use.

Since botnet mining exploits mostly unused resources and hence does not interfere with most other typical botnet activities, it can be expected that mining remains profitable for large botnets. But why break into other people's systems if we can use free resources? [Ragan and Salazar 2014] asked this question and used free trials and freemium accounts of cloud services to develop a cloud-based mining botnet. The main challenge was to automate the process, especially generating "credible" email addresses, to acquire a significant amount of bots, i. e., free accounts. In their tests, they were able to aggregate computing power worth thousands of dollars per week.

## 4.6 Bitcoin-inspired Network Applications

Apart from revolutionizing the area of digital currencies, Bitcoin also inspired other applications, most notably many network applications. Examples include but are not limited to a decentralized domain name system [vinced 2011], abuse prevention of cloud services [Szefer and Lee 2013], decentralized cloud storage [Wilkinson et al. 2014], or anonymous and distributed messaging [Warren 2012]. In the following, we will highlight two examples which build upon a fundamental insight into the role and properties of the block chain.

Since the early days of computer networks, naming services played an important role. Generally speaking, naming services map keys to values. DNS, which translates domain names to IP addresses, is probably the most well-known one.

Zooko Wilcox-O'Hearn conjectured [Wilcox-O'Hearn 2010] that when designing a naming service, one can choose only two out of the three properties "distributed", "secure",

and "human-meaningful". Examples include OpenPGP public key fingerprints, which are secure and decentralized, but not human meaningful. Domain names, in contrast, are meaningful and can be considered secure, but are managed centrally. The conjecture that building a system incorporating all three properties is infeasible became known as Zooko's triangle.

Bitcoin breathed new life into the feasibility discussion. The late Aaron Schwartz described a naming service based on Bitcoin's protocol [Schwartz 2011], which defies Zooko's triangle. He leverages the decentralized block chain as a key-value storage. Instead of assigning coins to addresses, he proposes to translate meaningful names to addresses. In his design, the proof-of-work scheme secures the mapping stored in the block chain in the very same way in which Bitcoin secures the transactions. Roughly at the same time, a Bitcoin-based naming service was discussed in [kiba 2010], and only a few months later Namecoin [vinced 2011] was announced. The authors of [Barok 2011] also provide additional background information on the development history.

Namecoin (NMC, `namecoin.info`) is an alternative approach to DNS, coordinating .bit domains. It shares the codebase with Bitcoin and inherits its properties. The mining process is therefore identical to Bitcoin, but starts with a fresh genesis block and consequently creates on its own block chain. For the most part, Namecoin extends the Bitcoin protocol to handle additional information (such as domain names) and introduces three new types of transactions: `name_new`, `name_firstupdate` and `name_update`. Assume Alice wants to register example.bit. First, she needs to broadcast a special pre-order transaction of type `name_new`. It consist of a sufficiently high network fee (currently 0.01 namecoins) as input and a `name_new` output script which includes the encrypted domain name. Please note: at this point the domain is not yet owned by Alice or anyone else. Thus, it is still possible to issue another `name_new` transaction with the same domain name, for example when loosing the necessary key to successfully connect to the output script. After a mandatory waiting period of 12 blocks, Alice broadcasts the actual registration in a `name_firstupdate` transaction. It publicly announces the domain name in plaintext and assigns the ownership. Updates need to be performed every 36,000 blocks (approx. 250 days) at the latest by issuing a `name_update` transaction, otherwise the domain expires.

The initial pre-ordering prevents others from quickly registering the same domain name when seeing the registration. The 12 blocks waiting period provides time to broadcast the registration and to anchor it in the block chain. The network fee's purpose is to prevent from massive pre-ordering. Once used in a pre-order, the fee gets destroyed and cannot be used for normal payments anymore. Thus, domain names are, in a sense, attached to "special" coins, which can, however, still be exchanged and traded through updates. Indeed, updates are basically normal transactions referring to the previous update. They enable, for instance, IP address updates and domain transfers. In Namecoin, every user can become her own domain registrar. However, the system is not strictly limited to domains. For example CertCoin [Fromknecht et al. 2014] proposes a authentication system based on Namecoin.

In principle, any type of data can be registered (also in Bitcoin), as long as it follows the protocol specification. So-

```
scriptPubKey: OP_RETURN <data>
```

Script 4: Null data standard transaction script template.

called *Null Data transactions* (cf. Script 4) allow to encode small, arbitrary data into the block chain. Interestingly, [Bos et al. 2013] found values which do not appear to correspond to valid cryptographic key pairs; they seem to encode ASCII characters.

Bitmessage [Warren 2012] takes the idea a step further and implements an anonymous, distributed, and encrypted messaging protocol. The protocol specification is quite different from Bitcoin (and Namecoin), but the source of inspiration is clearly visible. First of all, there is no block chain, because it is not a design goal to store all messages forever. Instead, Bitmessage is considered a best-effort service, which asks peers to save messages for two days only. In order to be sure that messages are successfully received, an acknowledgment mechanism is implemented. If an acknowledgment is missing, the sender re-broadcasts the message with exponential backoff intervals. Before broadcasting a message, the sender needs to provide a valid proof of work with the message. This is very similar to Hashcash [Back 2002] and limits spam and DoS attacks. The difficulty is adjusted according to the message size. Like transactions and blocks, messages are flooded through an unstructured overlay network. This mixes all circulating messages of all users, making it difficult to link sender and receiver. In addition, messages are encrypted with the recipient's public key and have no visible addresses attached. Therefore, every peer needs to decrypt every received message to check if it is the intended recipient.

# 5. PRIVACY

The original Bitcoin paper briefly describes privacy considerations: in contrast to traditional banking—that is, trusted third party models which limit the accessible trading information—Bitcoin's block chain publicly reveals all transaction data. The public addresses in the block chain, though, intend to provide pseudonymity, so that this openness of the transaction history does not automatically imply identifiability. To support this feature, a new key pair (and thus a new address) should be used for each transaction. In this sense, Bitcoin clients use so called *change addresses* (sometimes also called *shadow addresses*) by default, which are generated for each transaction and receive the change of the transaction on the output side.

However, as [Nakamoto 2008a] already points out and as we know from privacy and social network research [Narayanan and Shmatikov 2009, Backstrom et al. 2007, Narayanan and Shmatikov 2008], even when hiding behind multiple pseudonyms, these can be linked and often reveal identifying information. Along these lines, there is a significant body of research on the analysis of Bitcoin's publicly available block chain [Ron and Shamir 2013, Reid and Harrigan 2013, Androulaki et al. 2012, Meiklejohn et al. 2013, Ober et al. 2013, Baumann et al. 2014, Ron and Shamir 2014, Vasek and Moore 2015].

In the following, we will outline the methodology most block chain analysis approaches follow (Sec. 5.1), before

we then summarize interesting results obtained thereby (Sec. 5.2). Subsequently, we describe current practices and research insights when it comes to increase Bitcoin's level of privacy (Sec. 5.3).

## 5.1 Methodology of Block Chain Analyses

The reader might have noticed from previous sections that in Bitcoin there is no such thing like a "from" attribute in a transaction. A single, isolated transaction does not state an originator. Transactions refer to previous outputs only. By following this reference, we may infer something from their destination and consider it as originator. Consider transaction $t_3$ from Figure 6a for example, which sends coins to the address $a_6$ (assume $a_9^*$ is the change address). From the destination address $a_4$ of transaction $t_1$ we may say that $t_3$ is "from" $a_4$. Thus, even though block explorers often display a "from" attribute as if it was encoded in the transaction they truly infer it from the "last sent-to" address. Since transactions can have numerous inputs and outputs (as it is the case for $t_3$), it is, strictly speaking, not even a single "last sent-to" address but rather it should be termed "all last sent-to" addresses. Therefore, in order to conduct an analysis of the block chain, pre-processing steps are necessary, including the construction of several graphs, namely transaction, address and entity graphs. Let us take a closer look at these steps (which are illustrated in Figure 6) and the possibilities to link Bitcoin users.

### *Transaction Graph.*

The most straightforward step is to construct a *transaction graph* $\mathcal{T}(T,L)$, where $T$ is the set of transactions in the block chain and $L$ is the set of directed assignments (i. e., transaction output-input relations) between these transactions. Each assignment $l \in L$ carries a number of coins $C_l$. Inherently, transactions have a total order defined by the block chain, so, as [Reid and Harrigan 2013] noted, there cannot be any loops in $\mathcal{T}$.

### *Address Graph.*

From the assignments in the transaction graph, we can infer an origin-destination pair of Bitcoin addresses. This will be possible at least for most standard transactions such as P2PKH. Based on these relations, we can derive an *address graph* $\mathcal{A}(A,L')$, where $A$ is the set of all Bitcoin addresses and $L'$ is the set of directed assignments, but this time connecting addresses rather than transactions. Optionally, we can make $\mathcal{A}$ a multigraph and add a timestamp as an attribute to each $l \in L'$ so as to distinguish between multiple assignments between the same pair of addresses. Please note that some assignments will not yield an origin-destination pair. For example, coinbase blocks have no origin address (i. e., no input) and point to one destination address (i. e., one output) only.

### *Entity Graph.*

The next step aims at grouping addresses which *probably* belong to the same user. Based on a number of heuristics, which are either directly derived from the Bitcoin protocol or reflect common practices, a so-called *entity graph* can be constructed. The entity graph $\mathcal{E}(E,L'')$ consists of a set $E$ of entities, where each $e \in E$ is a disjoint subset of addresses $A$. Like [Ron and Shamir 2013], we use the neutral term entity here to express the possibility of errors and the missing ground-truth knowledge about real ownerships.

The widely most accepted heuristic is to assume that all input addresses of a given transaction belong to the same entity. Based on this heuristic, we can cluster the transitive closure of this property over all transactions. If, for example, $a_1$,$a_2 \in A$ are the inputs of one transaction $t_1 \in T$ and $a_2$,$a_3 \in A$ of another transaction $t_2 \in T$, then we may conjecture that $a_1$,$a_2$,$a_3$ all belong to the same entity.

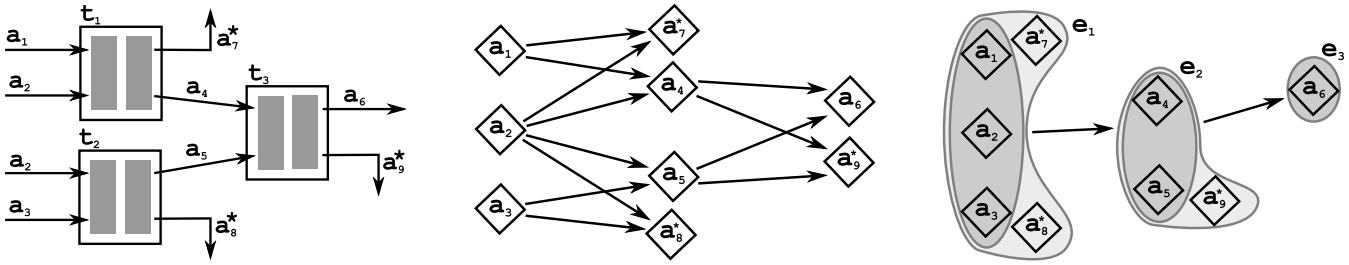
In [Androulaki et al. 2012], a second heuristic is introduced, which also clusters completely *new* addresses on the output side to the input entity, assuming that such an address is the change address. In particular, consider two output addresses $a_i^*$,$a_j \in A$ where $a_i^*$ is an address which never before appeared in the block chain and will never be re-used to receive payments, and $a_j$ is an address which was part of at least one previous transaction. In this case, $a_i^*$ is assumed to be the change address and belongs to the same entity as the input addresses. The authors of [Meiklejohn et al. 2013] argue that due to mining pools or gambling sites, it is common to issue transactions to multiple different users with probably more than one new address. Hence they refined the heuristic, adding the condition that it should only be applied if there is only one *single* new address $a_i^*$ in the transaction. Furthermore, they also excluded self-changes and coinbase transactions from the clustering.

In general, these and other heuristics take advantage of typical idioms of use and thus are prone to errors in case of unconventional Bitcoin applications. False positives in the clustering process can join addresses into huge entities which actually do not belong together. This has been observed in [Meiklejohn et al. 2013], leading the authors to further refinements. Mostly by manual inspection they identified usage patterns involving services such as SatoshiDice. SatoshiDice sends payouts back to the same address. If a user spent coins from a change address in a gamble, the address would receive another input which invalidates the one-time receive property of a change address. They used this observation to clean up their heuristic from false positives.

Independent from this particular case, idioms of use constantly change. For example, the community recommends not only to use a fresh address as the change address, but also a fresh receiving address. The rule of thumb is: use a fresh address whenever possible. However, there is also an approach with the opposite intention, i. e., enriching transactions with a so-called *marker address* (a. k. a. *green address*) to link the transaction to an entity [Vornberger 2012]. This and related proposals [Vandervort 2014] can be used to leverage from existing trust relationships so as to accept transactions quicker, for example. In addition, new techniques such as CoinJoin [Maxwell 2013a], where different users join in a single transaction, alter the usage patterns. Thus, heuristics must consider these changes, i. e., they must constantly be refined and adapted. Some heuristics might even hold true for a segment of the block chain only, while a certain usage pattern was common.

### *Ownership.*

Mapping addresses and thus entities to identities finally requires side channels. Some addresses, e. g., from WikiLeaks or Silk Road, are publicly known. Many services, like online stores or exchanges, expect the user to identify before using the service. Others can be detected by using web crawlers searching social networks (like, for instance,




(a) Transaction Graph (b) Address Graph (c) Entity Graph

Figure 6: Block chain analysis.

bitcointalk.org) for Bitcoin addresses (in the users' post signatures, for example) as it was done in [Reid and Harrigan 2013, Fleder et al. 2013]. The software BitIodine [Spagnuolo et al. 2014] offers an automated analysis framework. Using similar means as described above, it parses the block chain, constructs the respective graphs, uses heuristics for clustering, and links addresses to users by adding side channel information.

Another approach is to match IP addresses. [Reid and Harrigan 2013] used publicly available data sets from Bitcoin faucets, which give out coins for free. These services save and publish the IP address of the recipients to prevent abuse. A more rigorous way is to exploit the information that can be revealed by observing the network. In particular, as explained in Section 4, it is possible to correlate transaction originators to IP addresses. Based on the techniques presented in this section, transaction can be linked with the Bitcoin address and thus also with the IP address. If the linked Bitcoin address is part of an entity cluster, the whole cluster with all involved transactions is deanonymized.

## 5.2 Block Chain Analysis Results

Based on the methods described above, a number of interesting and useful insights on the use of Bitcoin and the flow of transactions can be gained. In this section we will give a brief summary.

[Reid and Harrigan 2013] and [Fleder et al. 2013] identified transactions and entities of interest and illustrated their relationships. [Fleder et al. 2013] used the PageRank algorithm [Page et al. 1999] to find "important" entities: entities with more "references" (i. e., a higher number of incoming transactions) are ranked higher, as well as entities referenced by entities with a high rank. With this method, [Fleder et al. 2013] identified the addresses of SatoshiDice and the FBI as particularly interesting. [Reid and Harrigan 2013] started from known addresses such as WikiLeaks and visualized its neighborhood. This way they (directly or indirectly) linked entities to their donations and revealed that donations were forwarded to other addresses.

The graphs allow to track Bitcoins along a sequence of transactions and to reveal patterns. Popular strategies to divert large amounts of Bitcoins (e. g., from thefts) are to create long chains of transactions with branches and (re-)collections [Ron and Shamir 2013, Reid and Harrigan 2013, Meiklejohn et al. 2013]. Such a practice can be used to obscure Bitcoin flows. Noteworthy is a pattern termed *peeling chain* [Meiklejohn et al. 2013], which was observed in multiple studies: a single address starts with a large bitcoin amount. In a transaction, a small fraction is "peeled off" and transferred to a change address. This procedure is repeated, potentially hundreds of times. The peeled-off amounts often reside in *saving accounts* and often have never since been moved [Ron and Shamir 2013]. Sometimes, small amounts are aggregated to a large amount, forming the starting point for another peeling chain. Through careful analysis, it was possible to expose the meaningful recipient of the Bitcoin transactions. In a similar maneuver from a theft in 2011, [Reid and Harrigan 2013] revealed the attempt to lay a false trail. After the theft, Bitcoins were transferred to an address which is associated with the hacker group LulzSec. The thief most probably tried to draw the attention to somebody else; the analysis did not reveal evidence for any other relationship between LulzSec and the thief. More types of scams and their prevalence are investigated by [Vasek and Moore 2015].

Bitcoins residing in saving accounts are often called *dormant coins* [Ron and Shamir 2013, Ober et al. 2013]. They are moved only occasionally, if at all. One famous example is the wealth of Dread Pirate Roberts (DPR), the operator of Silk Road, the biggest online marketplace for drugs and much more (sometimes also "amazon.com of illegal drugs") The FBI was able to seize only a small fraction of his coins; the rest still resides in saving accounts [Ron and Shamir 2014].

In general, the degree of anonymity of an entity can be expressed through a set of entities within which it is indistinguishable. The bigger this so-called *anonymity set*, the stronger the anonymity. [Ober et al. 2013] assume that dormant entities, which are currently not active, do not increase the anonymity set. Thus, when taking a certain point in time into consideration the number of active entities are a better estimate.

Their analysis reveals a scale-free distribution (power law) of active days (i. e., days with cash flow) for the entities. Consequently, there is a large number of entities active for one day only, but also many single entities active for long periods. According to the findings, in order to blend into a large anonymity set, the aim is to create an entity as "small" as possible (i. e., with a small number of associated addresses), and to be active for a short time only.

The consensus of all aforementioned contributions is that there exist means to link information in Bitcoin's block chain. This makes Bitcoin not as private as it is often assumed. Quite in contrast: it is probably the most transparent trading system ever built. The block chain is a huge record, which enables everybody to evaluate all transactions.

## 5.3 Enabling Privacy

Even though privacy is not an inherent property of Bitcoin, it is strongly associated with it. Hence, it is often used for purposes where sender and/or receiver intend to remain anonymous. Dread Pirate Roberts, the operator of Silk Road, is cited in an interview after his arrest with the statement that Silk Road would not have been possible without Bitcoin [Greenberg 2013]. Besides, there is a strong desire to create an anonymous digital currency. In this section we discuss some of the best-practice techniques and proposals, which enable (more) privacy either within Bitcoin or in related digital currency systems.

In order to prevent block chain analysis techniques from succeeding, decoupling of information about the sender and the receiver is required. In analogy to mix networks for network anonymity like Tor (which, as discussed above in Sec. 4.4, decouple sender and receiver addresses in communication), mixing services for Bitcoin transactions—*money laundry services*—can be constructed. Indeed, there are many parallels between these fields.

The easiest way for gaining anonymity in Bitcoin is a third-party approach, comparable to a single-hop anonymization proxy in anonymous communication: a trusted third party gets the bitcoins (including a tip), with the request to transfer the coins to a given destination address. Due to the extremely large trading volume, the popular dice gamble SatoshiDice could leave the impression of being a good way to obscure transactions and to launder bitcoins (even when losing every now and then). As noted by [Meiklejohn et al. 2013], the winnings are tied to the wager transaction, though. Thus, a block chain analysis would be able to follow the flow. In order to avoid this, the coins need to be juggled around by the third party in such a way that incoming and outgoing transactions cannot be linked.

Indeed such services exist. They all follow similar principles: transactions are routed through a shared wallet, so as to break the chain of trackable transactions. Assume Alice wants to send a bitcoin to Bob. Alice sends the bitcoin to a new bitcoin address generated by the mixing service. The mixing service aggregates bitcoins received from all its users to re-distribute them again. Some services use randomness, i. e., they split the amount into smaller random chunks and transfer them after random intervals. The user can schedule the transactions and provide time constraints. This way, as indicated in Figure 7, rather than receiving the bitcoin directly from Alice, Bob will receive it from others, which ideally will not reveal a relationship between Alice and Bob. [Möser et al. 2013] experimentally analyzed three services, namely Bitcoin Fog (`bitcoinfog.com`), BitLaundry (`bitlaundry.com`) and Send Shared by `blockchain.info`[3]. They confirmed that most of them indeed obscure the transaction, but they still have issues and leak links. The most important reason is the sometimes low usage volume and, in consequence, the small anonymity set. If this happens and no reserve assets exist in the mixing services' pool, it will likely use the just-received bitcoins next.

Sometimes a *taint analysis* as provided by `blockchain.info`[4] is used to evaluate the anonymity provided by a mixing service. It describes which fraction

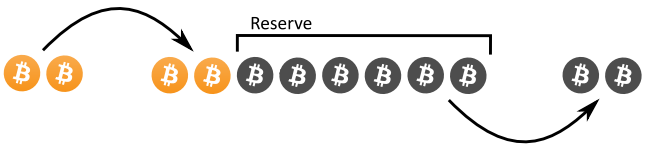



Figure 7: The principle of Bitcoin mixing services.

of coins received by a Bitcoin address can be traced back to another address. The more "tainted" the chain of transactions is, the stronger the linkage between the involved addresses remains. The only untainted coins are freshly minted coins from the coinbase transaction. However, taint analyses are not widely accepted, because they do not provide much information about the anonymity of a mixing service. Due to the lack of a good measure, [in Bitcoin 2015] define the so called *taint resistance*. They use the Matthews correlation coefficient (MCC) to express the accuracy an adversary obtains by linking inputs and outputs of a single transaction.

The discussion reminds of the early years of anonymous communication [Chaum 1981] and the subsequent research on mixing services [Danezis et al. 2003, Möller et al. 2003, Serjantov et al. 2002, Diaz and Serjantov 2003]. Aspects and techniques such as trust, observability, timing correlation, batching, dummy traffic and many more play a central role in this field just as in bitcoin mixing services. However, the case of Bitcoin reveals two significant differences: a major trust problem and a predominant global adversary. Both characteristics will be discussed in the next paragraphs.

With respect to trusting a single mix, the same holds for network anonymity and bitcoin mixing: the mix is always able to resolve the mapping between sender and receiver. But as long as the service is trustworthy and nobody else has access to their servers, plausible deniability remains. So, what to do if the mixing service is *not* trustworthy? The standard answer is: concatenate multiple mixes in a cascade. Even if individual mixes are compromised, as long as there is at least one honest mix, anonymity can be expected. (In theory at least: there exist attacks where it is sufficient to control a subset of mixes [Raymond 2000, Serjantov et al. 2002, Syverson et al. 2000].) However, the trust problem in Bitcoin has a dimension that does not exist in network anonymity: since Bitcoin deals with monetary values, trust also means to put stock in somebody else's hands. This is, of course not entirely different from data forwarding: data also has a value, especially when it includes passwords or personal information. Thus, data is typically secured by cryptographic means before handing them over to a mix. An adversary "running away with the encrypted data" can be considered acceptable, as long as the cryptography is strong enough: the encrypted data is worthless for an untrustworthy mix. From Bitcoin's perspective, transferring coins means changing the ownership in a irreversible way. At this point, the mix (who might be malicious) is—by the protocol—the legitimate owner of the coins. Thus, he could spend them for whatever he likes. This monetary aspect should not be underestimated, as it amplifies the trust problem with mixing services.

An adversary with a global view of the network has long been considered unlikely in the field of network anonymity. Even though it is now no longer considered that unlikely by

[3] The mixing service Send Shared was dropped by blockchain.info. Their new service Shared Coin follows a different strategy, which we will cover later in this section.

[4] `https://blockchain.info/taint/1dice6GV5Rz2iaifPvX7RMjfhaNPC8SXH`

all [Johnson et al. 2013], such an adversary remains a very strong assumption. Bitcoin, in contrast, inherently does have such global observers: the block chain is publicly available. In both network anonymity and Bitcoin anonymity we can distinguish between passive (honest but curious) and active adversaries, but the adversary models are different.

[Bonneau et al. 2014] proposes a mixing service design named Mixcoin[5]. The design is not entirely different from the previously deployed mixing services, but pays particular attention to the implications of parameter choices such as mixing and transaction fees in the face of different adversary models. If not considered carefully, fees add a "tag" (in a sense) to the transaction and help tracking the path. Mixcoin also includes a reputation-based approach for accountability, which is expected to reduce the probability of thefts. The authors of [Blindcoin Blinded 2015] extend the Mixcoin protocol by applying blind signatures [Chaum 1982] to prevent the mix from learning input-output mappings: users provide inputs and cryptographically blinded outputs. The mix signs the blinded outputs in return. Now, users can anonymously reconnect and reveal their "unblinded" output to the mix, which can verify that the outputs were signed in advance.

CoinJoin [Maxwell 2013a] is an approach which tries to go further in the sense that it aims to securely prevent theft, while at the same time providing privacy and being fully compatible with the Bitcoin protocol. It exploits the fact that multiple inputs of a transaction are independent from each other. For transaction verification, every input needs a valid signature, where the signatures do not necessarily need to stem from the same user. Thus, it is possible that users agree on a set of inputs and outputs and independently sign the transaction without the risk of theft. In order to blend in, all outputs in a CoinJoin transaction need to send the same amount of coins. The anonymity in a single transaction is limited by the number outputs, though, but can be significantly improved by concatenating CoinJoin transactions. Even relaxing the conditions and simply joining casually in transactions (i. e., without caring for the output volumes) impedes block chain analyses. In fact it breaks the fundamental heuristic from Section 5.1 (inputs of a transaction belong to the same entity). As another side effect, per-user transaction fees are reduced, making joint transactions cheap. However, the devil is in the detail: unless very carefully implemented, it is possible to link inputs and outputs easily [Atlas 2014, S. 2014, in Bitcoin 2015].

Besides, CoinJoin has two major weaknesses: (i) it needs to manage the collection of signatures, which involves the risk of additional privacy leaks, and (ii) participants can mount a denial-of-service attack by stalling joint transactions.

Regarding (i), users need to negotiate all transaction details in advance, construct the transaction, collect the respective signatures and finally broadcast the transaction in the bitcoin network. The negotiation could be done informally—on IRC, for example—or in a more structured way using a specialized distributed protocol. The easiest way, of course is to employ a central entity, i. e., a server, which takes control. This basic idea of CoinJoin is implemented and offered as an online service named Shared Coin (`sharedcoin.com`) where `blockchain.info` takes the role of the central instance. There is also a decentralized implementation for Bitcoin named Coinmux (`coinmux.com`). Darkcoin (DRK, `darkcoin.io`) is an altcoin which offers CoinJoin-like transactions natively [Duffield and Hagan 2014].

Independent from using a centralized vs. decentralized or a formal vs. informal way of negotiation, some instances will invariably learn the mapping of input and output addresses. In the decentralized case, all participants will know the mapping of their allies. In the centralized case, the server will know the mapping of all participants. As [Maxwell 2013a] sketched, blind signatures can help to solve this issue. First approaches in these directions can be found in [Ladd 2013].

CoinShuffle [Ruffing et al. 2014] is a decentralized approach which tackles the problem by forming a chain of participants and using layered cryptography in a similar manner as in mix networks [Chaum 1981]. CoinShuffle participants send their transaction destination—encrypted in layers—along the chain of participants. In each intermediate step, the respective participant removes a layer of encryption. Additionally, each participant adds his designated destination, likewise encrypted in layers. The last participant of the chain receives the full list of destinations from its predecessor, removes the final encryption layer from all of them, and finally adds his own destination. The transaction can now be constructed and signed as proposed by the CoinJoin protocol. Note that the last peer of the chain of participants cannot link individual destinations. Likewise, intermediate peers cannot link destinations because they are encrypted.

With respect to (ii) above, a user committing her intention to participate in a joint transaction, but later on not signing it, can stall the successful conclusion of the transaction. Blacklisting users deemed malicious may help, but comes with the bitter taste of false positives. However, ideas presented by [Saxena et al. 2014] might mitigate the problem: the authors develop a primitive called composite signatures, which is based on aggregate signatures [Boneh et al. 2003]. Initially composed of a masking key, it allows to incrementally add new signatures to the composite signature. The advantage over CoinJoin is that input and output addresses need not be known in advance. The composite signature can rather be passed around, making it more robust against DoS. Since the aggregation process is irreversible—that is, it is hard to compute individual signatures based on the composite signature—the approach provides plausible deniability. This requires modifications to the Bitcoin protocol, though. I. e., it changes the way signatures and references are computed and verified.

Approaches such as the fair exchange protocol by [Barber et al. 2012] and CoinSwap by [Maxwell 2013b] continue to reduce the necessary mutual trust and thus also enable anonymous peer-to-peer mixing. They make use of Bitcoin's scripting features, i. e., multi-signature and hash-locked transactions. A number of related Bitcoin-based commitment protocols can be found in [Andrychowicz et al. 2014a, Bentov and Kumaresan 2014]. The authors of [Bissias et al. 2014] support this trend and present a discovery mechanism by publishing "ads" in the block chain, which thwarts Sybil and DoS attacks.

The general idea of CoinSwap [Maxwell 2013b], for example, is that Alice and the mixing service build a 2-of-2 multi-signature transaction with Alice's coins. The mixing

[5] Please note that Mixcoin here refers to the respective work by Bonneau et al. and not to the identically named altcoin Mixcoin (MIX, `mixcoin.co`)

service and Bob build another 2-of-2 multi-signature transaction with the mixing service's coins. These transactions are announced publicly. The respective refund transactions are time-locked and held back for safety, in case one party vanishes. Next, Alice and the mixing service as well as Bob and the mixing service each construct and exchange redeem transactions for the multi-signature transactions. Both redeem transactions are additionally hash-locked by the *same* secret (which comes from Bob). Thus, both of them can be redeemed by the respective participants as soon as the secret is known. This ensures that if Bob gets paid, the mixing service gets paid, too. Once the mixing service becomes confident that it gets paid, it can release the multi-signature. The same applies to Alice. In the end, successful CoinSwap transactions are not distinguishable from standard multi-signature transactions.

The anonymity set of CoinSwap consists of all 2-of-2 multi-signature transaction published at roughly the same time. Since Alice could also play the role of Bob in the protocol, CoinSwaps can be chained. Every CoinSwap requires four transaction and rather complex staged phases. As mentioned, it can also be used to exchange (or mix) coins in a peer-to-peer fashion. CoinJoin, in comparison, is less complex, but also has a limited anonymity set, namely the number of participants in a single transaction. However, both approaches seem to have a place: CoinJoin could be used (opportunistically) to increase overall privacy for everyone, and CoinSwap could achieve stronger anonymity when desired, at the cost of additional transactions.

Instead of developing means of usage which increase the privacy, there are also approaches which extend the Bitcoin protocol or propose altcoins with native untraceable transaction support. The aforementioned Darkcoin [Duffield and Hagan 2014] is one example. Another approach builds upon non-interactive zero-knowledge proofs [Blum et al. 1988]. Zerocoin [Miers et al. 2013] is a protocol extension to Bitcoin by which Alice can prove to others that she owns a bitcoin and is thus eligible to spend *any* other bitcoin. First she produces a secure commitment, i. e., the zerocoin, which is recorded in the block chain so that others can validate it. In order to spend a bitcoin, she broadcasts a zero-knowledge proof for the respective zerocoin, together with a transaction. The zero-knowledge proof protects Alice from linking the zerocoin to her. Still, the other participants can verify the transaction and the proof. Instead of a linked list of Bitcoin transactions, Zerocoin introduces intermediate steps. Unfortunately, even though Zerocoin's properties may seem appealing, it is computationally complex, bloats the block chain and requires protocol modifications. However, it demonstrates an alternative, privacy-aware approach.

An extension of Zerocoin is presented by [Androulaki and Karame 2014]. The authors developed additional means to also hide the coin volume of transactions and Bitcoin addresses. A fully-fledged altcoin design named Zerocash with strong privacy guarantees was presented in [Ben-Sasson et al. 2014]. It takes the zero knowledge approach from Zerocoin, but improves it both in terms of functionality and efficiency.

Another altcoin approach is CryptoNote [van Saberhagen 2013], which denotes a protocol and technology framework. An actual implementation is Bytecoin (BCN, `bytecoin.org`). CryptoNote aims for the same two major privacy features as Zerocoin and its extensions: unlinkable transactions and untraceable payments. Unlinkable transactions help hiding the balance of a Bitcoin address. In Bitcoin, users are able to prevent this by always using a fresh address. However, as the existing results based on block chain analyses underline, if the public address is know (as with WikiLeaks or Silk Road, for example), the account balance can be determined. CryptoNote utilizes the basic idea of the Diffie-Hellman exchange protocol [Diffie and Hellman 1976] to derive one-time key pairs for each transaction. These key pairs are generated on demand by the respective parties, without previous interaction. In particular, the sender uses the recipient's public address and generates a one-time public key to which the coins are sent. In addition, the sender adds half of the Diffie-Hellman handshake to the transaction. The receiver can use this half and its original private key to compute the private key required to redeem the transaction.

In order to further increase the difficulty of tracing payments and coin flows, CryptoNote uses *ring signatures* [Rivest et al. 2001]. Ring signatures secure a message like any digital signature, but can be produced by any member of a group. Unlike group signatures, ring signatures make it infeasible to determine which member of the group signed the message. Every member can compute a ring signature on a message using their own private key and the group's public keys. For CryptoNote this means: when signing a transaction, in addition to the output he owns, the sender selects multiple other outputs of foreign transactions with the same amount (i. e., outputs she does not necessarily own). She then joins them as a single input. From the public keys of all outputs and her own private key, the sender creates the respective ring signature for the input. The validity of the transaction only implies that one of the group members has signed the transaction and spends a coin, but not which coin exactly. This is significantly different from Bitcoin, because only one of the selected outputs can actually be spent, not all of them. It increases the resistance against analysis of the block chain by providing multiple plausible paths to follow. Every consecutive transaction amplifies the effect by adding additional cash flow options.

However, such a scheme raises questions regarding double spending. In particular, if it is not possible to determine which coin has been spent, how can attempts of double spending be detected? CryptoNote tackles this issue by employing *traceable ring signatures* [Fujisaki and Suzuki 2007]: if somebody uses a private key more than once to create a signature, this can be detected, because every transaction also holds a so-called *key image*. If the same key image reappears, it means the one-time private key has been used more than once. This indicates double spending. Therefore, every peer keeps track of the previously seen key images. As with Zercoin, though, the increased privacy level comes at the price of a bloated block chain and more complex operations.

# 6. PROOF-OF-X (POX) SCHEMES

For attempts to paraphrase the Bitcoin protocol and to give a generalized description of its purpose, probably notions such as "consensus", "distributed" and "verification" come to mind. All of these are closely related to the role of proof of work in the Bitcoin design. In the following, we will dig deeper into this aspect. This exposes challenges which are sometimes much more subtle than the ones we saw before (like double spending and scalability issues), but

which likewise are of substantial importance. In response to these additional challenges, many alternative protocols have been proposed, forking from the Bitcoin protocol and implementing their own currency. We already mentioned examples such as Dogecoin, Darkcoin, and Bytecoin. But there are many more altcoins, and we will highlight a few in this section.

## 6.1 Reaching Consensus – The Byzantine Generals Problem

From a general perspective, Bitcoin works towards a consensus in a distributed manner and replicates the state network-wide. In order to guarantee fault tolerance, some redundancy is necessary. The amount of redundancy depends on the failure types. Consider, for example, three entities holding a value. Assume that a single entity fails: it always returns the same, wrong value. By comparing the answers, it is easy to identify the failing entity and to decide on the true value. Two entities would not suffice, an analogous situation would result in a conflict. In general terms, in the presence of $f$ failures, the network needs $n \geq 2f + 1$ entities to tolerate the failures.

However, failures in a broader sense can also be random or malicious. These failures are called Byzantine failures, after the famous Byzantine Generals problem [Lamport et al. 1982]. The original problem description considers the case of $n$ generals trying to mutually agree via messengers on a common battle plan. However, $f$ of the generals are traitors and try to thwart the agreement. The situation is comparable to a distributed system which aims to reach consensus. The Byzantine Generals problem with synchronous and reliable communication reaches consensus as long as $n \geq 3f + 1$ is satisfied [Lamport et al. 1982]. In case of asynchronous communication, the Fischer-Lynch-Paterson (FLP) proof shows that a asynchronous and deterministic consensus protocol cannot tolerate any failures at all [Fischer et al. 1985]. Later [Dolev et al. 1987] refined the results and revealed the dependency on the system properties process synchronicity, communication delay (bounded or undbounded), message order, and transmission method (point-to-point or broadcast). An overview of the first decade of this broad field is given by [Lynch 1989, Turek and Shasha 1992].

The topic kept research busy. Some contributions relaxed the conditions in order to overcome the impossibility results. Non-deterministic consensus protocols, for example, provide solutions to the asynchronous case [Aspnes 2003]. While consensus becomes possible due to the randomness, the approach leaves a (small) chance of failing. Others replaced the requirement of each entity to commit to a final decision by accepting that each entity has a current view (or "opinion") that may change as execution proceeds [Angluin et al. 2006]. This yields the concept of "stabilizing consensus".

All of this comes quite close to what we see in the Bitcoin protocol: it makes heavy use of randomness in the mining process and re-adjusts it regularly. Furthermore, Bitcoin deliberately omits a final and fixed ownership attribution[6]. Instead, it uses a rule of thumb: after (typically) six confirmations, transactions are considered settled. Convergence is achieved via the longest chain rule.

[6] By now, Bitcoin in fact has checkpoints, which are once in a while hardcoded into the software and mark an irreversible block height in the block chain. Checkpoints were introduced by Satoshi Nakamoto later and are not part of the original design.

A popular example which borrows directly from the results of Byzantine agreement is Ripple (XRP, `ripple.com`). Ripple takes a different direction and relies on a set of trusted "authorities" to build consensus. It implements a round-based consensus algorithm with final decision making. The final decision results in a so-called "last closed ledger" which represents the current state of all accounts. Ripple's consensus algorithm achieves $5f+1$ resilience [Schwartz et al. 2014].

Another condition is the timing model, which describes the message propagation in the network. In the synchronous communication case, messages arrive after a certain fixed time span. Any message that takes longer is considered a failure. Protocols for the synchronous case explicitly rely on timing assumptions; they often proceed in discrete rounds. As numerous impossibility results show, solutions are often only possible for the synchronous case. In the much more general asynchronous timing model, no assumptions are made on the relative rate of execution or message delivery. As [Aguilera 2010] points out, the asynchronous timing model might not be realistic, because it is unlikely that messages take longer than a certain threshold (even though this threshold could well be very high). However, protocols for the asynchronous case—where they exist—are very generic and work irrespective of the systems condition.

Thus, the system designer is left with two bad choices. The fact that Bitcoin allows the blocks' timestamp to deviate only within a certain range hints at the assumption of a synchronous network model. The 10 minutes block creation time was, in the first place, chosen as a tradeoff between confirmation time and the amount of work wasted due to chain forks. But it also ensures to reach every corner of the network even in the face of prolonged propagation times. It therefore, in a sense, synchronizes the network (loosely).

In addition to the already mentioned conditions, the Bitcoin network is, due to its structural properties which support anonymity, of unknown size. In the face of Byzantine failures, this additional condition makes the problem harder. Malicious entities can set up fake identities which subvert the election and inject faulty information, i. e., they can mount a Sybil attack [Douceur 2002]. In Bitcoin, the proof-of-work requirements tackle this vulnerability by artificially increasing the cost of a vote. This approach originates from the attempts to combat spam [Dwork and Naor 1993, Back 2002]. In fact, the idea to use it in the context of Byzantine agreement protocols has been proposed before Bitcoin already [Aspnes et al. 2005]. Yet, even though the possibility of Sybil attacks had been considered there, the size of the network was assumed to be known; otherwise, the assumptions are comparable to those behind Bitcoin.

Thus, in principle, all the puzzle pieces required to build a consensus protocol similar to Bitcoin were there. The similarity of the problem and the advances in the field were recognized [Szabo 2003]. The idea to employ Byzantine agreement protocols such as [Malkhi and Reiter 1998] to the area of distributed digital currencies paved the ground [Szabo 1998]. Even design proposals and prototypes existed [Dai 1998, Finney 2004, Szabo 2005]. Eventually, in 2008, time was ready and Bitcoin appeared.

In summary, we can say that Bitcoin tackles the Byzantine Generals problem from a practical angle. Nakamoto himself compared the protocol design to this particular problem [Nakamoto 2008c]. As also discussed in [Miller 2012], Bit-

coin balances viability and security and seems to have found a sweet spot. It suggests that Bitcoin underlies the assumptions of a synchronous network of unknown size, relaxes the deterministic constraint and takes eventual consistency as adequate. [Miller and LaViola Jr 2014] considered these assumptions and derived a fault tolerance of $2f + 1$, where $f$ is the total hash power of malicious miners (Byzantine failures). This meets the intuitions and the analyses of the 51% attacks: as long as more than half of the hash power is controlled by honest miners, the network will eventually reach consensus, even in the presence of malicious miners. However, as [Garay et al. 2014] add to this view, Bitcoin meets the theoretical fault tolerance only as long as the assumption of synchronicity holds. Thus, information propagation amongst honest peers is essential, especially when the malicious miners' hash power approaches the critical threshold; otherwise the system becomes fragile and insecure.

## 6.2 Proof of Work — the Monopoly Problem

Proof of work is a key component of Bitcoin. Inherently, any task suitable as a basis for proof-of-work schemes needs to be difficult to solve, but trivial to verify. It often boils down to a random process of trying to find a solution to a puzzle—like a (partial) hash collision.

Bitcoin's precursors B-money [Dai 1998], RPOW [Finney 2004] and Bit Gold [Szabo 2005] already incorporated a proof-of-work scheme in one way or the other. In all these cases, the motivation was to consider the solution to the puzzle as a scarce and valuable good, like "gold". RPOW is a centralized approach, which uses reusable proof of work as token money. Coins are minted by a server in return to a proof of work. The coins are reusable and transferable, while the central server checks for validity. B-money decentralizes the process by assuming a synchronous unjammable broadcast channel. Transactions are issued by signing a contract, which is broadcast so that everybody knows of it. Alternatively, a trusted set of servers can be employed to keep track of the ledger. Bit gold chains the proof of work and uses the last entry to create the next challenge. It uses a Byzantine agreement protocol which relies on a quorum of addresses rather than a quorum of computing power. It is thus vulnerable to Sybil attacks. Finally, it was the Bitcoin protocol to combine Sybil resistance and coin minting by a sophisticated proof-of-work scheme.

Originally, the paradigm of proof of work is "one-CPU-one-vote" [Nakamoto 2008a]. Bitcoin uses a CPU-bound function (i. e., SHA-256 [Eastlake and Hansen 2011]) as the basis for its proof-of-work scheme. Miners are by nature rational profit seekers. Their mining costs consist of expenses for mining hardware and ongoing energy cost. They strive to reach the break-even point as quickly as possible, to make as much profit as possible. The first miners used computers with ordinary CPUs to solve the proof of work. Even though CPUs are extremely versatile, the versatility comes at the expense of limited speed. Miners therefore quickly sought for faster solutions to dominate the competition and to make more profit.

Mining operations are highly parallelizable. Some graphics processors (GPUs) are therefore able to compute the repeated hash operations much faster and much more energy efficient than any CPU. GPU mining quickly replaced CPU mining. Bitcoin's popularity picked up pace and pushed the competition even more. The fact that CPU-bound hash functions are suitable for hardware acceleration [Chaves et al. 2008] received attention. Thus, it was only a matter of time until hardware-based mining solutions became available—first based on reconfigurable logic (Field-Programmable Gate Arrays, FPGAs) and subsequently based on application-specific integrated circuits (ASICs). FPGAs and especially ASICs significantly accelerate the speed and efficiency of mining. Since then, only ASICs are economically viable for bitcoin mining. They achieve hash rates in the order of one terahash per second. [Bedford Taylor 2013, O'Dwyer and Malone 2014] tell the story of Bitcoin hardware and its energy footprint in detail. Recent observations and optimizations [Courtois et al. 2014b] will probably continue to push the hash rates even higher in the future.

Following the increasing computational power, Bitcoin adjusts the difficulty, i. e., the target value, to maintain the ten-minute-per-block target rate. This behavior can be seen in Figure 8, which we generated by parsing the block chain and calculating a moving average of the block confirmation time (plotted on the left y-axis) for a small part of the block chain. We can observe a repeated decrease of the confirmation time, which implies that the total hash rate of all participants increases. Every 2016 blocks, Bitcoin adjusts the difficulty of the proof of work (plotted on the right y-axis). Only very seldom in the history of Bitcoin it happened that the difficulty was adjusted downwards. Most often it increases.

Overall the difficulty follows an exponential growth, as Figure 9 shows (please note the logarithmically scaled y-axes). Since the difficulty is continuously adjusted to the hash rate, the data line in this plot can be interpreted as either of these: the hash rate (according to the left y-axis) or the difficulty (according to the right y-axis).

The use of specialized mining equipment increases the voting power per entity. This development subverts the proof-of-work paradigm and therefore implies a threat. In particular, it reduces the democratic basis by suppressing "small" miners. As consequence, the trust in Bitcoin degrades.

In Bitcoin, we can take the idiom "rich gets richer" literally: it has been shown that the wealth of rich users increases faster than the wealth of users with low balance [Kondor et al. 2013]. Additionally, there is an alarming trend that the power of a small group of miners significantly exceeds the power that all other users contribute [Gervais et al. 2014]. This raises the question whether Bitcoin is still truly a decentralized currency or if it is shifting towards the centralized banking model which it originally questioned.

During the early steps of proof of work (i. e., Hashcash) and before Bitcoin, the imbalanced capabilities of systems were already identified as a possible issue. This was considered inherent to CPU-bound functions. In order to recover the situation, the idea to employ memory-bound functions instead was introduced. The approach is to incorporate large amounts of (unpredictable) memory access operations in the proof of work calculations, so that these constitute the dominant factor. Hence, solving the proof of work is limited by the memory access time, not by the CPU speed. The underlying assumption is that the differences between users with regard to memory access speed are inherently much smaller than the differences with respect to computing power. Memory-bound functions had been proposed in the context of spam prevention before [Dwork et al. 2003, Abadi et al. 2005].

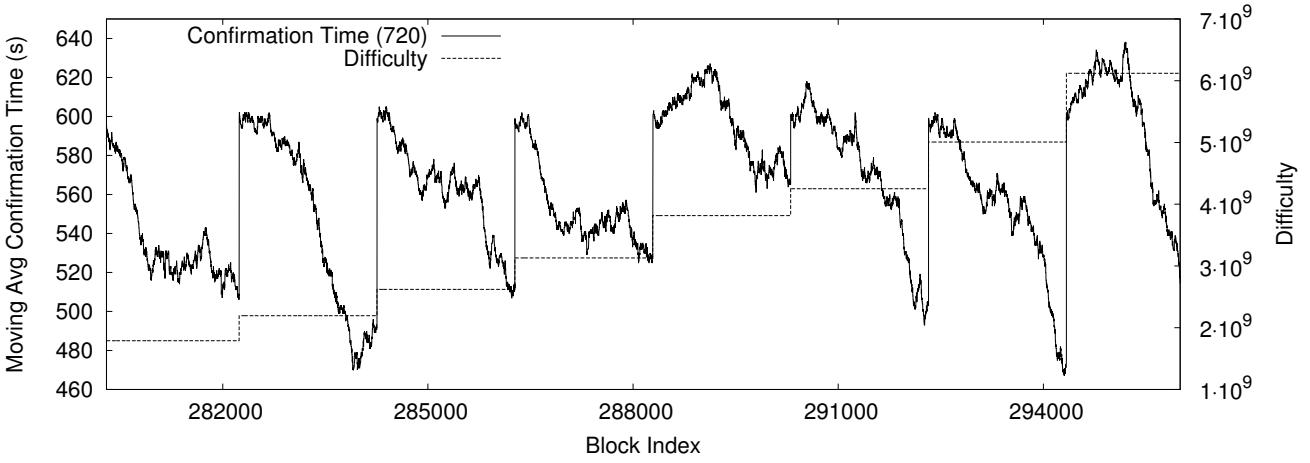



Figure 8: Moving average of the confirmation time and difficulty.

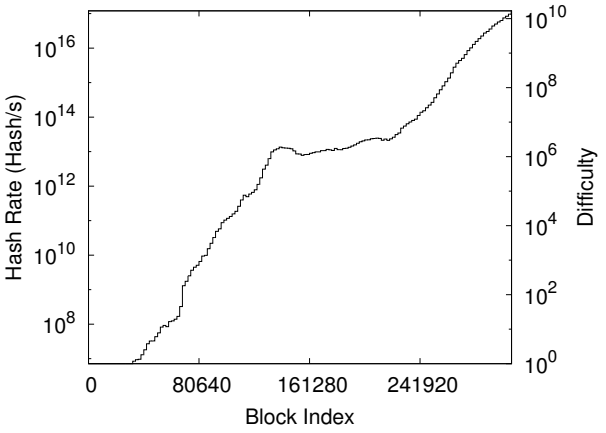



Figure 9: Hash rate and difficulty (logscale).

In the context of Bitcoin, functions such as scrypt [Percival 2009] and CryptoNight [van Saberhagen 2013] have been discussed. In a corresponding proof-of-work scheme, memory-intensive operations are added to the hashing operations. The intention is to foster more evenly distributed power among the user base, to avoid the emergence of a monopoly. Unfortunately, scrypt—which is probably the most popular alternative proof of work scheme, and is used, e. g., by Litecoin (LTC, `litecoin.org`)—still enables the use of specialized mining devices. The appearance of more energy-efficient GPU-based mining brought a significant increase of Litecoin's hash rate. Moreover, most alternative hash functions are rather immature and not as well analyzed as, for example, SHA-256. Therefore, concerns about possible future weaknesses remain. In general, there are discussions whether ASIC-resistant proof of work can exist at all [Hearn 2014, Lists 2014a].

Another fundamental criticism of proof of work in general and Bitcoin in particular is that it wastes computational power (and thus energy) without any intrinsic value. Contributions which try to alter this situation are NooShare [Coventry 2012], Primecoin [King 2013b] and Permacoin [Miller et al. 2014]. NooShare proposes the scheduling of arbitrary Monte-Carlo simulations as a proof of work. Primecoin (XPM, `primecoin.io`) requires miners to find long chains of prime numbers, so-called Cunningham chains. Permacoin realizes distributed storage by requiring so-called proofs of retrievability, that is, access to local storage. All three approaches provide an added value besides securing the block chain. However, other difficulties arise then, such as fine-tuning the difficulty and prohibiting the reusability of earlier results. For example, in case of Primecoin, the length of a prime chain does not naturally imply a suitable difficulty metric. Chains of eight primes may be a hundred times harder to find than chains of seven primes. Primecoin solves this issue by using Fermat's primality test, to construct an approximately linear difficulty metric for a given chain length. Even though Primecoin is considered a CPU-only currency (since finding prime chains appears to be inefficient on GPUs), doubts along the same lines as mentioned earlier exist [Koooooj 2013], and first GPU-based miners are already available.

For completeness, we briefly mention some altcoins which also rely on proof of work. We have already mentioned Litecoin as an adopter of scrypt as the underlying hash function. Dogecoin (DOGE, `dogecoin.com`) is another popular altcoin which uses scrypt. Both are Bitcoin forks (and thus use the same code base), but have faster confirmations times of 2.5 minutes and 1 minute, respectively. Dogecoin also has a much higher coin supply and a higher reward per block. Therefore, it is gaining traction as a microdonation system.

Not deployed, but nevertheless appealing is the idea of FawkesCoin [Bonneau and Miller 2014]. As the name indicates, it builds upon the Guy Fawkes signature protocol [Anderson et al. 1998], which substitutes Bitcoin's elliptic curve DSA (ECDSA). FawkesCoin thereby constructs a digital currency with only symmetric cryptography. Due to the symmetry property, signatures can be used securely only once. Hence, FawkesCoin requires to use a fresh address for every transaction (which, however, is also recommended for Bitcoin anyway).

## 6.3 Proof of Stake — Towards Solving Incentive Problems

Even though Bitcoin vividly shows that a digital currency based on proof of work is viable, weaknesses still exist. The participants of Bitcoin pay the miners via a mechanism of inflation to secure the currency. Nevertheless, the future when the block reward declines over time remains unclear. Approximately every four years (i. e., every 210,000 blocks) the block reward halves. After the first reward halving, we can observe a slight dip in the difficulty, implying that miners left the network. If the controlled supply of coins continues as specified, approximately in the year 2032 the reward will be less than 1 BTC, and in the year 2140 it will be down to zero. According to [Courtois 2014], this kind of deflation is a self-destruction mechanism. It puts the security of crypto currencies at risk by driving off miners. Whether the transaction fees will suffice to compensate the decreasing reward and to provide the necessary incentive for miners remains unclear and is controversially discussed [Kroll et al. 2013].

The relationship between the miners' strategy and the mining reward follows a trade-off, which can be modeled using game theory [Houy 2014a]. Once the miner finds a solution, she needs to make sure to propagate the information before others claim the next block to get the reward. The transactions to include in a block are chosen by the miner. We can assume that their number has no effect on the complexity of the proof of work, but that a higher number increases the time to reach consensus: the more bytes the block contains, the more time it takes to broadcast it through network. And the more transactions a block holds, the more time it takes to verify its validity. On the other hand, the more transactions there are in a block, the bigger the reward. According to [Houy 2014a], there is a unique Nash equilibrium, i. e., a point where no participant can gain an advantage from changing her strategy. Interestingly, at this Nash equilibrium miners will not include any transaction at all in their blocks. Obviously, this would render Bitcoin practically useless, which hurts miners as well. The Nash equilibrium shifts as soon as transaction fees increase or the block reward significantly decreases. This suggests that transaction fees might provide an incentive.

However, there is a problem with decreasing rewards, which boils down to the tragedy of the commons [Hardin 1968]. Tragedy of the commons is a game theoretic term which describes the phenomenon that taking individually, independently optimized (and thus most likely selfish) actions reduce the peer group's long-term gain by depleting a common resource. Known examples come from areas such as environmental pollution, fishing resources and road traffic.

The relevance to Bitcoin is multifaceted [Bentov et al. 2014]. From the users' perspective, it is in the interest of all users to provide an incentive for miners and to pay transaction fees in order to maintain a secure network. However, each individual user's (selfish) interest is to let others pay the fees. Users might therefore start to issue transactions without fees. If the majority acts this way, mining becomes unprofitable, and miners will give up. From the miners' perspective, there is another tragedy of the commons problem. On one hand, their group interest is to have users pay high fees per transaction. On the other hand, the cost of including a transaction in a block is negligibly small. Thus, each miner gains a personal advantage by simply including every transaction with a fee, independent from the actual amount. This might lead to continuously lower fees, up to the point where mining is not lucrative anymore. In both cases, miners will turn away and thus gaining a monopoly becomes easier. Again, there is a risk for the currency's security.

Questioning proof of work as a basis and looking for alternatives lets us revisit the fundamental requirements which need to be fulfilled: first, the block generation must be somehow "expensive", and individual miners shall not be able to gain an overproportionally high ability to mint coins. Second, consensus must eventually be reached; there must be a common rule to resolve forks and to determine the main block chain. Third, it must be forgery proof. The latter is a general requirement for currencies, which must hold here, too.

It turns out that *coin age* is a viable alternative to proof of work. Coin age is defined as the currency amount times the holding period [King and Nadal 2012]. For example, if Alice transfers two coins to Bob and Bob held the coins for 90 days, the coin age is 180 coin-days. When Bob spends the two coins, the coin age he accumulated is destroyed.

The idea to use the coin age to define the reward is known as *proof of stake* [King and Nadal 2012] (PoS). It is, for example, implemented in Peercoin (PPC, `peercoin.net`). Mining a proof-of-stake block requires to construct a so-called *coinstake block* (named after Bitcoin's coinbase transaction). In a coinstake transaction, owners send coins in their possession to themselves and add a predefined percentage as their reward. Analogue to proof of work, a hash value below or equal to a target value is required to successfully mint a block. In contrast to proof of work (and Bitcoin), the difficulty is individually determined: it is inversely proportional to the coin age. Because the hash is—except for a timestamp—calculated on static data, there is no way for miners to use their computational power to solve the puzzle faster than others. In particular, there is no nonce which can be modified. Instead, every second the timestamp changes and miners have a new chance of finding the solution. If they find a solution, they broadcast the block including the coinstake transaction. The coinstake transaction assigns the reward to the miner, but also resets the coin age. Of course, new coin age can subsequently be accumulated again, slowly increasing the chances of solving the puzzle next time.

One can think of coin age in proof of stake the same way as of computing power in proof of work. A huge pile of old coins is equivalent to a powerful ASIC mining rig. But there are key differences: the "power" is independent from computing power. Instead, it depends on the deposit. Thereby, proof of stake provides an answer to the criticism that proof of work wastes energy. Unlike Primecoin, which tries to put an intrinsic value into the proof of work, proof of stake eliminates the high energy consumption altogether. It shifts from a highly competitive tournament to a raffle-like scheme, with repeatedly occurring new chances for all participants. In addition, miners destroy the coin age by claiming the reward; they do not keep it for the next round. This gives others the chance to "win the raffle", too.

These properties mitigate the risk of monopoly in the tragedy of the commons problem. Note that the voting power is more equally distributed. Thus, "rich gets richer" is complemented by "poor gets richer" [King 2013a], meaning every participant can provide a proof of stake, thus help to secure the block chain and in return get a reward in proportion to their holding.

Besides, exploiting a proof of stake-based currency seems much more expensive. In contrast to proof of work, where the longest block chain survives, proof of stake declares the block chain with the highest total sum of destroyed coin age as the main chain. In order to perform an attack similar to the 51%-attack, an attacker must hold a huge amount of coins, which even when destroying the coin age suffices to gain more than half of the odds. It is assumed that the cost for gaining the majority of computing power in a proof-of-work scheme (e. g., for hardware) is smaller than the cost for buying enough coins in proof-of-stake setting. However, there are also objections to this claim [Houy 2014b], suggesting that the attack would be anticipated and thus coins would be ditched, which reduces the attacker's costs. Nevertheless an attacker holding lots of coins would suffer severely from ruining the currency, which probably reduces the incentive to do so in the first place.

For practical purposes, proof of stake is proposed to complement proof of work and to become the dominant factor when the proof of work block rewards subside. Indeed, Peercoin and others (such as Memcoin2 [Mackenzie 2010]) are designed as hybrid systems. Yet, there are also entirely proof of stake-based altcoins, such as Nextcoin (NXT, `nextcoin.org`).

Derivatives—such as transactions as proof of Stake (TaPoS) [Larimer 2013] and delegated proof of stake (DPoS) [Larimer 2014]—are supposed to mitigate the monopoly problem, and at the same time to make the system more secure by collecting votes from a larger base. We will cover the details of the problems which motivated these (and other) directions in the next section.

## 6.4 Proof of Activity — Incentivize Active Participation

Proof of stake shows an alternative to proof of work, but it comes with a number of limitations. And actually there is another tragedy of the commons problem, which affects proof of stake likewise.

[Babaioff et al. 2012] emphasize the goal of information propagation in Bitcoin. They compare it to the 2009 DARPA Network Challenge [DARPA 2009]: on the date of the 40th anniversary of the Internet, DARPA announced a challenge in which competing participants tried to find red weather balloons across the US. MIT's winning strategy [Pickard et al. 2010] consisted of recruiting hunters and offering a reward. However, they recognized that this is another challenge by itself, so they offered an additional reward for recruiters of balloon finders. This way they created an incentive to spread the word. In fact, the additional reward is necessary because each additional hunter competes with other hunters in the proximity, i. e., each additional participant reduces the others' chances of finding the balloon.

From the miners' perspective, the situation in Bitcoin is not different, especially when the block reward decreases and the transaction fees should compensate. This reveals another tragedy of the commons [Bentov et al. 2014]: a miner has an incentive to keep any transaction including a fee for itself. Eventually, the miner will not relay the transaction, so as to reduce competition. Instead of rewarding transaction fees to the miners, Peercoin destroys the fees in order to eliminate the incentive to not cooperate [King and Nadal 2012].

But proof of stake has some more limitations: it entirely depends on the coin age, which can be accumulated by holding coins only, and can be claimed in a coinstake transaction to oneself. Coins spent in regular transactions assigning coins to others also destroy the coin age, but are not considered in the proof-of-stake raffle. Thus, hoarding coins is encouraged; [Ren 2014] therefore talks of *collectibles* rather than currencies.

The major weakness, though, is that coin age accumulates even when the node is not connected to the network. It suffices when nodes come online occasionally and wait for their reward, only to go offline again afterwards. This behavior will result in a more bursty reward distribution than in the case where nodes remain online all the time, but stakeholders most likely will accept that. The lack of a sufficient number of online nodes, though, facilitates attacks.

Any reward scheme which tries to incentivize in one or the other sense must pay attention to Sybil attacks. For example, MIT's strategy in the DARPA challenge was vulnerable, because balloon hunters could set up false identities as recruiters to increase their finder's reward. [Babaioff et al. 2012] provide first findings in this direction. In the following, we summarize two extensions [Ren 2014, Bentov et al. 2014] which incorporate proof of stake and provide an advanced reward scheme. Related approaches which avoid the usage of proof of stake exist, too [Paul et al. 2014].

[Ren 2014] follows the idea that a higher activity produces a healthier economy. The author identifies the problem that coin age is a linear function of time. In practice, Peercoin implements an upper and a lower bound of coin age to mitigate some of the mentioned problems. However, changing the increment function to an exponential decay function, for example, would have a profound impact. In such a setting, the increment rate of the coin age decreases with time and asymptotically converges to zero. Parameterizing the decay constant allows for a deliberate specification of the function's half-life time. This changes the incentive: a fresh coin accumulates coin age much faster, up to a fixed value. The intention is to reduce the resistance to trade coins and to encourage users to stay online. It is conceivable to employ other functions, such as non-monotonic and/or periodic functions. That would imply to punish hoarding coins even more, or to reflect a seasonal pattern. The basic idea is termed *proof of stake velocity* and is implemented in Reddcoin (RDD, `reddcoin.com`).

The approach by [Bentov et al. 2014] is to directly reward active peers for their contribution. The idea is to raffle a fraction of the proof-of-work block reward among all active nodes, while their stake determines the amount of raffle tickets, i. e., their chances of winning. It is thus a combination of proof of work and proof of stake. In detail, miners mine "empty" block only. If they solve the proof-of-work puzzle, they broadcast it in the network as before. Everybody receiving the block derives $N$ deterministic pseudorandom ticket numbers from it. The first $N-1$ most lucky stakeholders sign the block with their respective private key and broadcast the signature. If the $N$-th most lucky stakeholder sees the block, she creates a wrapper, includes the block, all transactions, the $N-1$ signatures, adds her own signature and broadcasts the wrapped block. Others will consider it as a legitimate extension of the block chain if the block and the lucky stakeholders are valid. Finally the transaction fees are shared by stakeholders and the miner.

In order to determine the $N$ lucky stakeholders, the pseudorandom number is interpreted as an index in the list of all so far minted satoshis. Finding the user in possession of the satoshi is done by a procedure called "follow-the-satoshi". Everybody can verify it by inspecting the block chain and following the satoshi from the coinbase transaction up to the address currently holding it. Please note that this is much like proof of stake, with the difference that the coin age is irrelevant. Alice holding two coins has twice as high a chance to be picked as Bob holding one coin. The decision to share the transaction fees as reward among the stakeholders and not, for example, the entire block reward is rooted in the observation which we pointed out earlier: a high reward for the stake incentivizes undesirable coin hoarding. The small fees are considered a nice bonus, but not an incentive for hoarding.

The underlying concept is to reward active peers which are online. If one of the lucky stakeholders is offline, he will not be able to respond and to add her signature. Hence, the block cannot be completed. At some point, there will be another miner solving the proof of work, drawing $N$ different stakeholders. The difficulty is adjusted according to the hash rate and the fraction of active peers. The concept is accordingly called *proof of activity* (PoA) [Bentov et al. 2014]. It rewards stakeholders who participate rather than punishing passive stakeholders. The proof-of-work component is inevitable to make the system converge and to provide a rule to resolve forks. Moreover, it is necessary to throttle the speed of picking the lucky stakeholders. Proof of activity improves security: besides a huge amount of computational power, an attacker needs a significant amount of stake to double spend.

## 6.5 Proof of Publication — Provable Commitments

After looking at various proof-of-X schemes, let us close the circle and come to another achievement of computing history which apparently influenced the Bitcoin design, namely secure timestamping. A timestamping service provides timestamps for digital documents, which securely keep track of the creation and modification time of the document. There are many timestamping schemes, such as PKI-based centralized services, where documents and timestamps are hashed and secured by the private key of the timestamping server. However, the server can easily backdate documents by hashing and signing a previous timestamp. Thus, the approach comes with the premise of trusting the timestamping server.

In order to tackle this issue, [Haber and Stornetta 1991] developed so-called linked timestamps. Each timestamp certificate includes a hash of the previous one. This ensures a total order of documents, even if inaccurate clocks are in place. Furthermore, it hardens fake certificates, because it is not possible to retroactively hook documents in the linked chain of timestamps.

As [Haber and Stornetta 1991] already note, it is possible to distribute the process of secure timestamping. Instead of relying on a single server, it is favorable to consult multiple instances. The hash of the document can be interpreted as a $k$-tuple of server IDs to request certificates from. However, a vulnerability to Sybil attacks shows up again, and the problem in general is very closely related to the Byzantine Generals.

Once again, this is where Bitcoin steps in: Bitcoin also references previous transactions and links blocks, but it also provides Sybil resilience. In fact, it is essential for a transaction system to determine the order of actions. Since Bitcoin's proof-of-work mechanism constantly readjusts the difficulty to meet the target of one block every 10 minutes, the protocol can also be considered a distributed secure timestamping service, where the timestamp accuracy is roughly the block generation time.

The authors of [Clark and Essex 2012] come to a similar conclusion and propose a carbon dating commitment protocol based on Bitcoin, namely CommitCoin. The idea is, in a more general context, also known as *proof of publication* and comes in various manifestations. In the case of CommitCoin, a Bitcoin address is generated which encodes the information of a respective document. Other use cases include coin tosses or lotteries [Back and Bentov 2014, Andrychowicz et al. 2014b], where a group of players bet coins by issuing elaborate transactions. Finally one player is chosen randomly as the winner and gets the money. This works without a central entity and without the need to trust each other. Generalizations of these protocols are provided by [Andrychowicz et al. 2014a, Kumaresan and Bentov 2014].

Proofs of publication require a number of complex steps to encode the data and to preserve the involved coins. However, intentional coin destruction can also be used for sound purposes. At the beginning, especially with currencies relying on proof of stake, the problem of how to bootstrap a new currency exists. Bitcoin, for example, (and many other proof of work-based currencies) had its genesis block. Other altcoins, such as Counterparty or Mastercoin[7], bootstraped by using the idea of a *proof of burn*. By "burning" coins of one currency in a verifiable but unspendable way, coins of another currency can be generated and assigned. At first sight, burning coins might seem to be a harsh primitive of commitment, but it can be considered as expensive, just like a proof of work.

This property makes proof of burn a very well viable tool for migration. Simple protocol changes such as invalidating an old transaction type might involve a *soft fork*, but generally remain backward compatible. Older clients will continue to accept transactions which are considered invalid by new software releases. In order to enforce the changes, the majority of the miners must upgrade. Miners that do not upgrade may waste computing power by generating blocks that will be rejected by others. On the other hand, miners have a handle to oppose unwilling upgrades as long as the majority refuses.

Changes to core components such as the block structure, difficulty rules, or the set of valid transactions are much more difficile and often involve a *hard fork*. Since it makes previously invalid transactions or blocks valid, it requires all users and miners to upgrade. However, as it was done with P2SH transactions, changes could be implemented in such a way that new transaction types appear to older clients like a previously already valid transaction type. If this is not possible, proof of burn steps in and provides a way of moving from one chain to another. The authors of [Back et al. 2014] take it one step further: why not send coins

[7] Strictly speaking, Mastercoin did not use proof of burn. It rather used a so-called exodus address, which was a regular Bitcoin address and funded the development of the currency. On a more abstract level, though, it follows the same idea.

not only to addresses within one block chain, but also to concurrent block chains? As a result, they developed so-called side chains. Similar to proof of burn, coins are sent to a special output. Thereby, they are not destroyed, but only "immobilized" until somebody can prove they are no longer being used elsewhere. This adds a form of reversibility to the proof of burn. In order to make this work for Bitcoin, a new validation rule would have to be introduced, though, which would require at least a soft fork.

Similar to proof of burn, provable commitment protocols such as proof of bandwidth or proof of retrievability can be used to repay expenses as it was proposed in the context of Tor [Ghosh et al. 2014, Jansen et al. 2014, Biryukov and Pustogarov 2014b] or to realize a decentralized file storage [Miller et al. 2014].

# 7. SUMMARY OF OBSERVATIONS AND FUTURE RESEARCH DIRECTIONS

After our extensive characterization of Bitcoin and its related approaches, it is time to take a look back and briefly summarize our observations, before we discuss future research directions and the next generation of Bitcoin.

We exposed the agreement on and maintenance of an unforgeable, but distributed, ledger—the block chain—which holds all assignments ever created—the transactions—and makes them verifiable to everyone. Bitcoin is considered a secure timestamping service and a practical solution to the Byzantine Generals problem. In order to achieve consensus, Bitcoin accepts the risk of failure. In particular, double spends (or race attacks) are and will always be possible. The risk, though, can be minimized and is subject to a personal trade-off.

The transparency of the system is a fundamental aspect to achieve verifiability, but likewise it introduces an omnipresent global attacker model. Therefore, we can conclude Bitcoin is anything but private. Nevertheless, it can hide identities and the efforts to strengthen this property continue. The attempts and techniques recall the discussion on anonymity networks. However it also brings up commitment schemes such as zero knowledge proofs and gives them a whole new "playground". Thus, we see the starting point of a symbiotic effect between different areas.

Probably one of the most outstanding contributions of Bitcoin is the degree of decentralization which was previously deemed impossible. The ingenious concept of mining, may it be based on proof of work, proof of stake or something else, secures the ledger and eventually stabilizes the consensus. It binds "votes" to something "expensive" and incentivizes to "pay" for it by holding out the prospect of a reward, which at the same time controls the money supply of the digital currency. Without mining, fake identities would be able to subvert the consensus and destroy the system. Because of this crucial point, we can conclude that $> 50\%$ attacks are the worst-case scenario. Furthermore, the mining monopoly threatens the decentralization.

The previously listed observations and features are often considered to be the most innovative parts of Bitcoin. But only in combination with one of the most distinctive features, namely the scripting capabilities, the depth of possibilities becomes evident. The instruction set may be limited, but it is still a powerful tool to realize sophisticated transactions and contracts. This trend is further followed by a second generation of cryptocurrencies with a fully-fledged scripting language. Yet the possibilities are barely explored.

For the future it remains unclear whether Bitcoin can and will stay as robust as it is today. Especially the scalability of the network and the subsiding rewards are pressing and need to be addressed. At the moment both apparently work well enough. In case of the peer-to-peer network, though, one can already observe symptoms of degradation, such as a long-tail distribution of the information propagation delay. However, Bitcoin's security assumptions rely heavily on the fast propagation of transactions and blocks. Thus, in order to scale to more participants and higher transaction rates the network needs to be able to handle the load. In case of the subsiding mining rewards the research community is unsure whether this poses a real problem or if fees are able to provide the necessary incentive. So far, we discussed a huge body of approaches which aim to be a solution to issues in Bitcoin. Some of them are deployed as an altcoin or an additional service. We provide a summary in Table 2, where we also point to the respective papers and the section(s) where we discussed them. Yet it remains unclear which alternative approach is most promising to actually improve Bitcoin, and which will survive in practice.

All the altcoins can be considered as a huge testing environment, from which Bitcoin can borrow in the future to address weaknesses. Bitcoin is constantly evolving and remains under development. It established the so-called Bitcoin improvement proposal (BIP) design documents as a way to introduce new features to Bitcoin. If a BIP finds the mutual consent of the community, it becomes accepted. However, since some of the proposed changes are more invasive than others, migration is an important point.

In summary, Bitcoin and the huge zoo of altcoins constitute a highly dynamic and certainly not yet fully understood field of research, in which numerous open questions with very high practical relevance remain to be answered. It will certainly be highly interesting to follow the future developments in this dynamic field.

# 8. CONCLUSION

In this survey we studied the broad field of Bitcoin, its characteristics and related concepts. In particular, we investigated the protocol's foundations, including the role of proof of work, and their relationship to security and network aspects. By doing so, we provided a holistic technical perspective on distributed currencies—and we pointed to manyfold open research opportunities.

# 9. ACKNOWLEDGMENTS

The authors would like to thank Daniel Cagara for the discussions on this topic and for sharing his practical experience.

# 10. REFERENCES

[Abadi et al. 2005] Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. 2005. Moderately Hard, Memory-bound Functions. *TOIT: ACM Transaction on Internet Technology* 5, 2 (May 2005), 299–327.

[Aguilera 2010] Marcos K Aguilera. 2010. Stumbling over consensus research: Misunderstandings and issues. In *Replication*. Springer, 59–72.

[Ali et al. 2015] Syed Taha Ali, Patrick McCorry, Peter Hyun-Jeen Lee, and Feng Hao. 2015. ZombieCoin: Powering

| | Approach | Distinct Feature (incl. References) | Sec. |
|---|---|---|---|
| **Precursor** | B-Money | Mining reward proportional to proof of work difficulty; requires a broadcast channel [Dai 1998] | 2.2, 6.1, 6.2 |
| | Bit Gold | Chained proof of work secured by timestamping [Szabo 2005]; Byzantine-resilient quorum to prevent double spending [Szabo 1998] | 3.2, 6.1, 6.2 |
| | RPOW | Centralized (reusable) proof of work exchange/ bank [Finney 2004] | 6.2 |
| **Altcoins** | Bitshares (BTS) | Delegated proof of stake [Larimer 2014] | 6.3 |
| | Bytecoin (BCN) | Implements CryptoNote [van Saberhagen 2013], which aims for unlinkable transactions and untraceable payments | 5.3, 6.2 |
| | Counterparty (XCP) | Colored coin; used proof of burn | 6.5, 6.5 |
| | Cryptonite (XCN) | Implements the mini block chain scheme [Bruce 2014] | 4.3 |
| | Darkcoin (DRK) | Implements native CoinJoin-like transactions [Duffield and Hagan 2014] | 5.3 |
| | Dogecoin (DOGE) | Block payload holds TXIDs only; fast block generation | 4.3, 6.2 |
| | Litecoin (LTC) | Uses scrypt [Percival 2009] to foster distributed power among miners | 6.2 |
| | Mastercoin (MSC) | Colored coin; exodus address | 6.5 |
| | Nextcoin (NXT) | Entirely proof of stake based | 6.3 |
| | Peercoin (PPC) | Identified coin age as alternative measure; proof of stake [King and Nadal 2012] | 6.3 |
| | Primecoin (XPM) | Proof of work with intrinsic value i. e. prime chains [King 2013b] | 6.2 |
| | Reddcoin (RDD) | Proof of stake velocity [Ren 2014] | 6.2 |
| | Ripple (XRP) | Implements a novel Byzantine agreement protocol [Schwartz et al. 2014] | 6.1 |
| | Zerocash | Full-fledged altcoin, carrying on the ideas of Zerocoin [Ben-Sasson et al. 2014] | 5.3 |
| **Protocols / Extensions** | CoinJoin | Uses multi-signature transactions to enhance privacy [Maxwell 2013a] | 5.3 |
| | CoinShuffle | Decentralized protocol to coordinate CoinJoin transactions [Ruffing et al. 2014] | 5.3 |
| | CoinSwap | Enables P2P-based trustless mixing [Maxwell 2013b] | 5.3 |
| | CommitCoin | Secure timestamping protocol [Clark and Essex 2012] | 6.5 |
| | Mini block chain | [Bruce 2014] identifies individual block chain components and proposes a mini block chain with a decentralized balance sheet | 4.3 |
| | Mixcoin | Mixing with accountability [Bonneau et al. 2014] | 5.3 |
| | Zerocoin | Unlinkable transactions and untraceable payments by employing zero knowledge proofs [Miers et al. 2013] | 5.3 |
| **Altchains** | Bitmessage | Secure messaging service [Warren 2012] | 4.6 |
| | Ethereum (Ether) | Turing complete smart contract processing [Wood 2014, Buterin 2014] | 2.5 |
| | Namecoin (NMC) | Key-value storage; realizes decentralized domain name coordination [vinced 2011] | 4.6 |
| | Metadisk | Decentralized file storage [Wilkinson et al. 2014] | 4.6 |

Table 2: Summary of altcoins and extensions.

Next-Generation Botnets with Bitcoin. In *BITCOIN '15: Proceedings of the 2nd Workshop on Bitcoin Research.*

[Anderson et al. 1998] Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. 1998. A new family of authentication protocols. *ACM Operating Systems Review (SIGOPS)* 32, 4 (1998), 9–20.

[Andresen 2011a] Gavin Andresen. 2011a. BIP 11: M-of-N Standard Transactions. (Oct. 2011). Status: Accepted.

[Andresen 2011b] Gavin Andresen. 2011b. BIP 13: Address Format for pay-to-script-hash. (Oct. 2011). Status: Final.

[Andresen 2012] Gavin Andresen. 2012. BIP 16: Pay to Script Hash. (Jan. 2012). Status: Final.

[Androulaki et al. 2012] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2012. Evaluating User Privacy in Bitcoin. *IACR: Cryptology ePrint Archive* 2012 (2012). Issue 596.

[Androulaki and Karame 2014] Elli Androulaki and Ghassan O Karame. 2014. Hiding Transaction Amounts and Balances in Bitcoin. In *TUST '14: Proceedings of the 7th International Conference on Trust & Trustworthy Computing.* 161–178.

[Andrychowicz et al. 2013] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2013. How to deal with malleability of BitCoin transactions. *CoRR: Computing Research Repository* (2013). `http://arxiv.org/abs/1312.3230`

[Andrychowicz et al. 2014a] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2014a. Fair Two-Party Computations via Bitcoin Deposits. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research.* 105–121.

[Andrychowicz et al. 2014b] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. 2014b. Secure Multiparty Computations on Bitcoin. In *SP '14: Proceedings of the 35th IEEE Symposium on Security and Privacy.* 443–458.

[Angluin et al. 2006] Dana Angluin, Michael J. Fischer, and Hong Jiang. 2006. Stabilizing Consensus in Mobile Networks. In *DCOSS '06: Proceedings of the 2nd International IEEE Conference on Distributed Computing in Sensor Systems.* 37–50.

[Asokan et al. 1997] Nadarajah Asokan, Phillipe A Janson, Michael Steiner, and Michael Waidner. 1997. The state of the art in electronic payment systems. *IEEE Computer* 30, 9 (1997), 28–35.

[Aspnes 2003] James Aspnes. 2003. Randomized protocols for asynchronous consensus. *Distributed Computing* 16, 2-3 (2003), 165–175.

[Aspnes et al. 2005] James Aspnes, Collin Jackson, and Arvind Krishnamurthy. 2005. *Exposing computationally-challenged Byzantine impostors.* Technical Report.

[Atlas 2014] Kristov Atlas. 2014. CoinJoin Sudoku. (2014). `http://www.coinjoinsudoku.com`

[Babaioff et al. 2012] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. 2012. On bitcoin and red balloons. In *EC '13: Proceedings of the 13th ACM Conference on Electronic Commerce.* 56–73.

[Back 2002] Adam Back. 2002. Hashcash-A Denial of Service Counter-Measure. (Aug. 2002). `http://www.hashcash.org/papers/hashcash.pdf`

[Back and Bentov 2014] Adam Back and Iddo Bentov. 2014. Note on fair coin toss via Bitcoin. *CoRR: Computing Research Repository* (Feb. 2014). `http://arxiv.org/abs/1402.3698`

[Back et al. 2014] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. *Enabling blockchain innovations with pegged sidechains.* Technical Report. `http://www.blockstream.com/sidechains.pdf`

[Backstrom et al. 2007] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. 2007. Wherefore Art Thou R3579x?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *WWW '07: Proceedings of the 16th International World Wide Web Conference.* 181–190.

[Bahack 2013] Lear Bahack. 2013. Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft). *CoRR: Computing Research Repository* (2013). `http://arxiv.org/abs/1312.7013`

[Bamert et al. 2013] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. 2013. Have a snack, pay with Bitcoins. In *P2P '13: Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing.* 1–5.

[Bamert et al. 2014] Tobias Bamert, Christian Decker, Roger Wattenhofer, and Samuel Welten. 2014. BlueWallet: The Secure Bitcoin Wallet. In *STM '14: Proceedings of the 10th International Workshop on Security and Trust Management.* 65–80.

[Barber et al. 2012] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. 2012. Bitter to better — how to make bitcoin a better currency. In *FC '12: Proceedings of the 16th International Conference on Financial Cryptography and Data Security.* 399–414.

[Barok 2011] Dušan Barok. 2011. *Bitcoin: censorship-resistant currency and domain system for the people.* Technical Report. Networked Media, Piet Zwar Institute, Rotterdam, Netherlands.

[Baumann et al. 2014] Annika Baumann, Benjamin Fabian, and Matthias Lischke. 2014. Exploring the Bitcoin Network. In *WebDB '04: Proceedings of the 10th International Conference on Web Information Systems and Technologies.*

[Bedford Taylor 2013] Michael Bedford Taylor. 2013. Bitcoin and the age of bespoke silicon. In *CASES '13: Proceedings of the 8th International Conference on Compilers, Architecture, and Synthesis for Embedded System.* 1–10.

[Ben-Sasson et al. 2014] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from Bitcoin. In *SP '14: Proceedings of the 35th IEEE Symposium on Security and Privacy.*

[Bentov and Kumaresan 2014] Iddo Bentov and Ranjit Kumaresan. 2014. How to Use Bitcoin to Design Fair Protocols. In *CRYPTO '14: Proceedings of the 34th Annual Conference on Advances in Cryptology.* 421–439.

[Bentov et al. 2014] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. 2014. Proof of Activity: Extending BitcoinâĂŹs Proof of Work via Proof of Stake. In *NetEcon '14: Proceedings of the 9th Workshop on the Economics of Networks, Systems and Computation.*

[Biryukov et al. 2014] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. 2014. Deanonymisation of Clients in Bitcoin P2P Network. In *CCS '14: Proceedings of the 21st ACM Conference on Computer and Communications Security.* 15–29.

[Biryukov and Pustogarov 2014a] Alex Biryukov and Ivan Pustogarov. 2014a. Bitcoin over Tor isn't a good idea. *CoRR: Computing Research Repository* (2014). `http://arxiv.org/abs/1410.6079`

[Biryukov and Pustogarov 2014b] Alex Biryukov and Ivan Pustogarov. 2014b. Proof-of-Work as Anonymous Micropayment: Rewarding a Tor Relay. *IACR: Cryptology ePrint Archive* 2014 (2014). Issue 1011.

[Bissias et al. 2014] George Dean Bissias, A. Pinar Ozisik, Brian Neil Levine, and Marc Liberatore. 2014. Sybil-Resistant Mixing for Bitcoin. In *WPES '14: Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society.* 149–158.

[Bitcoin dev 2014] Bitcoin dev 2014. Bitcoin Developer Documentation. (2014). `https://bitcoin.org/en/developer-documentation`

[Bitcoin wiki 2014] Bitcoin wiki 2014. The Bitcoin Wiki. (2014). `https://en.bitcoin.it/wiki`

[Blindcoin Blinded 2015] Accountable Mixes for Bitcoin Blindcoin Blinded. 2015. Luke Valenta and Brendan Rowan. In *BITCOIN '15: Proceedings of the 2nd Workshop on Bitcoin Research.*

[Bloom 1970] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

[Blum et al. 1988] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-interactive Zero-knowledge and Its Applications. In *STOC '88: Proceedings of the 20th Annual ACM Symposium on Theory of Computing.* 103–112.

[Boneh et al. 2003] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT '03: Proceedings of the 22nd International Conference on the Theory and Applications of Cryptographic Techniques.* 416–432.

[Bonneau and Miller 2014] Joseph Bonneau and Andrew Miller. 2014. FawkesCoin: A cryptocurrency without public-key cryptography. In *SPW '14: Proceedings of the 22nd International Workshop on Security Protocols.*

[Bonneau et al. 2015] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. 2015. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *SP '15: Proceedings of the 36th IEEE Symposium on Security and Privacy*.

[Bonneau et al. 2014] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2014. Mixcoin: Anonymity for Bitcoin with accountable mixes. In *FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security*.

[Bos et al. 2013] Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. 2013. Elliptic Curve Cryptography in Practice. *IACR: Cryptology ePrint Archive* (2013). https://eprint.iacr.org/2013/734.pdf

[Bruce 2014] J.D. Bruce. 2014. The Mini-Blockchain Scheme. (July 2014). http://cryptonite.info/files/mbc-scheme-rev2.pdf Rev. 2.

[Buterin 2014] Vitalik Buterin. 2014. A next-generation smart contract and decentralized application platform. (2014). https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf

[Chaum 1982] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *CRYPTO '82: Proceedings of the 2nd Conference on Advances in Cryptology*. 199–203.

[Chaum 1981] David L. Chaum. 1981. Untraceable Electronic Mail, Return addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (Feb. 1981).

[Chaves et al. 2008] Ricardo Chaves, Georgi Kuzmanov, Leonel Sousa, and Stamatis Vassiliadis. 2008. Cost-efficient SHA hardware accelerators. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16, 8 (2008), 999–1008.

[Chawathe et al. 2003] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. 2003. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 407–418.

[Clark and Essex 2012] Jeremy Clark and Aleksander Essex. 2012. Commitcoin: Carbon dating commitments with bitcoin. In *FC '12: Proceedings of the 16th International Conference on Financial Cryptography and Data Security*. 390–398.

[Cohen 2003] Bram Cohen. 2003. Incentives build robustness in BitTorrent. In *P2PEcon '03: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*. 68–72.

[Cohen 2014] Bram Cohen. 2014. An attack on the timestamp semantics of Bitcoin. (Nov. 2014). http://bramcohen.com/2014/11/03/an-attack-on-the-timestamp-semantics-of-bitcoin

[corbixgwelt 2011] corbixgwelt. 2011. Timejacking & Bitcoin. (May 2011). http://culubas.blogspot.de/2011/05/timejacking-bitcoin_802.html

[Courtois 2014] Nicolas T. Courtois. 2014. On The Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies. *CoRR: Computing Research Repository* (2014). http://arxiv.org/abs/1405.0534

[Courtois and Bahack 2014] Nicolas T Courtois and Lear Bahack. 2014. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. *CoRR: Computing Research Repository* (2014). http://arxiv.org/abs/1402.1718

[Courtois et al. 2014a] Nicolas T. Courtois, Pinar Emirdag, and Daniel A. Nagy. 2014a. Could Bitcoin Transactions Be 100x Faster?. In *SECRYPT '14: Proceedings of the 11th International Conference on Security and Cryptography*. 426–431.

[Courtois et al. 2014b] Nicolas T Courtois, Marek Grajek, and Rahul Naik. 2014b. Optimizing SHA256 in Bitcoin Mining. In *CSS '06: Proceedings of the 3rd International Conference on Cryptography and Security Systems*. 131–144.

[Coventry 2012] Alex Coventry. 2012. *NooShare: A decentralized ledger of shared computational resources*. Technical Report. http://web.mit.edu/alex_c/www/nooshare.pdf

[Dai 1998] Wei Dai. 1998. B-Money. (1998). http://www.weidai.com/bmoney

[Danezis et al. 2003] George Danezis, Roger Dingledine, and Nick Mathewson. 2003. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *SP '03: Proceedings of the 24th IEEE Symposium on Security and Privacy*. 2–15.

[DARPA 2009] DARPA. 2009. DARPA Network Challenge. (2009). http://archive.darpa.mil/networkchallenge/

[Decker and Wattenhofer 2013] Christian Decker and Roger Wattenhofer. 2013. Information propagation in the bitcoin network. In *P2P '13: Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing*. 1–10.

[Decker and Wattenhofer 2014] Christian Decker and Roger Wattenhofer. 2014. Bitcoin Transaction Malleability and MtGox. *Computing Research Repository (CoRR)* (2014). http://arxiv.org/pdf/1403.6676.pdf

[Dev 2014] Jega Anish Dev. 2014. Bitcoin mining acceleration and performance quantification. In *CCECE '14: Proceedings of the 27th IEEE Canadian Conference on Electrical and Computer Engineering*. 1–6.

[Diaz and Serjantov 2003] Claudia Diaz and Andrei Serjantov. 2003. Generalising Mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. 18–31.

[Diffie and Hellman 1976] Whitfield Diffie and Martin E Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976), 644–654.

[Dingledine et al. 2014] Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. 2014. One Fast Guard for Life (or 9 months). In *HotPETs '14: 7th Workshop on Hot Topics in Privacy Enhancing Technologies*.

[Dingledine et al. 2004] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security '04: Proceedings of the 13th USENIX Security Symposium*.

[Dmitrienko et al. 2014] Alexandra Dmitrienko, David Noack, Ahmad-Reza Sadeghi, and Moti Yung. 2014. On Offline Payments with Bitcoin (Poster Abstract). In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*. 159–160.

[Dolev et al. 1987] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. 1987. On the minimal synchronism needed for distributed consensus. *Journal of the ACM (JACM)* 34, 1 (1987), 77–97.

[Donet et al. 2014] Joan Antoni Donet Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartı. 2014. The Bitcoin P2P network. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Douceur 2002] John Douceur. 2002. The Sybil Attack. In *IPTPS '02: Proceedings of the 1st International Workshop on Peer-to-Peer Systems*.

[Drainville 2012] Danielle Drainville. 2012. *An Analysis of the Bitcoin Electronic Cash System*. Technical Report. University of Waterloo, Canada.

[Duffield and Hagan 2014] Evan Duffield and Kyle Hagan. 2014. *Darkcoin: Peer-to-Peer Crypto-Currency with Anonymous Blockchain Transactions and an Improved Proof-of-Work System*. Technical Report. http://www.darkcoin.io/downloads/DarkcoinWhitepaper.pdf

[Dwork et al. 2003] Cynthia Dwork, Andrew Goldberg, and Moni Naor. 2003. On memory-bound functions for fighting spam. In *CRYPTO '03: Proceedings of the 23rd Annual International Cryptology Conference*. 426–444.

[Dwork and Naor 1993] Cynthia Dwork and Moni Naor. 1993. Pricing via processing or combatting junk mail. In *CRYPTO '93: Proceedings of the 13th Conference on Advances in Cryptology*. 139–147.

[Eastlake and Hansen 2011] D. Eastlake, 3rd and T. Hansen. 2011. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234 (Informational). (May 2011). http://www.ietf.org/rfc/rfc6234.txt

[Elahi et al. 2012] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. 2012. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *WPES '12: Proceedings of the ACM Workshop on Privacy in the Electronic Society*.

[Eyal 2014] Ittay Eyal. 2014. The Miner's Dilemma. *CoRR: Computing Research Repository* abs/1411.7099 (2014). http://arxiv.org/abs/1411.7099

[Eyal and Sirer 2014] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In *FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security*.

[Feld et al. 2014] Sebastian Feld, Mirco Schönfeld, and Martin Werner. 2014. Analyzing the Deployment of Bitcoin's P2P Network under an AS-level Perspective. In *ANT '14: Proceedings of the 5th International Conference on Ambient Systems, Networks and Technologies*. 1121–1126.

[Feller 1957] Willliam Feller. 1957. *An introduction to probability theory and its applications*.

[Finney 2004] Hal Finney. 2004. RPOW. (2004). http://cryptome.org/rpow.htm

[Finney 2011] Hal Finney. 2011. Best practice for fast transaction acceptance - how high is the risk? (2011). https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384

[Fischer et al. 1985] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.

[Fleder et al. 2013] Michael Fleder, Michael Kester, and Sudeep Pillai. 2013. *Bitcoin Transaction Graph Analysis*. Technical Report. Massachusetts Institute of Technology (MIT), Computer Systems Security (6.858). http://css.csail.mit.edu/6.858/2013/projects/mfleder-mkester-spillai.pdf

[Fromknecht et al. 2014] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. 2014. *CertCoin: A NameCoin Based Decentralized Authentication System*. Technical Report. Massachusetts Institute of Technology, MA, USA. 6.857 Class Project.

[Fujisaki and Suzuki 2007] Eiichiro Fujisaki and Koutarou Suzuki. 2007. Traceable ring signature. 181–200.

[Gallagher and Kerry 2013] P Gallagher and C Kerry. 2013. Federal Information Processing Standards (FIPS) publication 186-4: Digital Signature Standard (DSS). (July 2013). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[Garay et al. 2014] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. 2014. The Bitcoin Backbone Protocol: Analysis and Applications. *IACR: Cryptology ePrint Archive* 2014 (2014). Issue 765. http://eprint.iacr.org/2014/765

[Gerhardt and Hanke 2012] Ilja Gerhardt and Timo Hanke. 2012. Homomorphic payment addresses and the pay-to-contract protocol. *arXiv preprint arXiv:1212.3257* (2012). http://arxiv.org/pdf/1212.3257v1.pdf

[Gervais et al. 2014] Arthur Gervais, Ghassan Karame, Srdjan Capkun, and Vedran Capkun. 2014. Is Bitcoin a Decentralized Currency?. In *SP '14: Proceedings of the 35th IEEE Symposium on Security and Privacy*.

[Ghosh et al. 2014] Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. 2014. A TorPath to TorCoin: Proof-of-Bandwidth Altcoins for Compensating Relays. In *HotPETs '14: 7th Workshop on Hot Topics in Privacy Enhancing Technologies*.

[Gnutella v0.4 2003] Gnutella v0.4 2003. The Annotated Gnutella Protocol Specification v0.4. (2003). http://rfc-gnutella.sourceforge.net/developer/stable/

[Goldfeder et al. 2014] Steven Goldfeder, Joseph Bonneau, Edward W Felten, Joshua A Kroll, and Arvind Narayanan. 2014. Securing Bitcoin wallets via threshold signatures. (2014). http://www.cs.princeton.edu/~stevenag/bitcoin_threshold_signatures.pdf

[Goldschlag et al. 1996] David M Goldschlag, Michael G Reed, and Paul F Syverson. 1996. Hiding Routing Information. In *IHW '01: Proceedings of the 1st International Workshop on Information Hiding*.

[Greenberg 2013] Andy Greenberg. 2013. An Interview With A Digital Drug Lord: The Silk Road's Dread Pirate Roberts (Q&A). (Aug. 2013). http://www.forbes.com/sites/andygreenberg/2013/08/14/an-interview-with-a-digital-drug-lord-the-silk-roads-dread-pirate-roberts-qa/

[Haber and Stornetta 1991] Stuart Haber and W. Scott Stornetta. 1991. How to Time-stamp a Digital Document. *Journal of Cryptology* 3 (1991), 99–111.

[Hajdarbegovic 2014] Nermin Hajdarbegovic. 2014. Bitcoin Miners Ditch Ghash.io Pool Over Fears of 51% Attack. (Jan. 2014). http://www.coindesk.com/bitcoin-miners-ditch-ghash-io-pool-51-attack/

[Hardin 1968] Garrett Hardin. 1968. The tragedy of the commons. *Science* 162, 3859 (1968), 1243–1248.

[Hearn 2014] Mike Hearn. 2014. Mining Decentralisation: The Low Hanging Fruit. (July 2014). https://bitcoinfoundation.org/2014/07/03/mining-decentralisation-the-low-hanging-fruit/

[Hearn and Corallo 2012] Mike Hearn and Matt Corallo. 2012. BIP 37: Connection Bloom filtering. (Oct. 2012). https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki

[Heilman 2014] Ethan Heilman. 2014. One Weird Trick to Stop Selfish Miners: Fresh Bitcoins, A Solution for the Honest Miner.. Poster. In *FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security*.

[Heusser 2013] Jonathan Heusser. 2013. *SAT solving - An alternative to brute force bitcoin mining*. Technical Report. https://jheusser.github.io/2013/02/03/satcoin.html

[Hoepman 2007] Jaap-Henk Hoepman. 2007. Distributed double spending prevention. In *SPW '07: Proceedings of the 15th International Workshop on Security Protocols*. 152–165.

[Houy 2014a] Nicolas Houy. 2014a. *The Bitcoin mining game*. Technical Report. http://halshs.archives-ouvertes.fr/halshs-00958224

[Houy 2014b] Nicolas Houy. 2014b. It will cost you nothing to 'kill' a Proof-of-Stake crypto-currency. *Economics Bulletin* 34, 2 (2014), 1038–1044.

[Howgrave-Graham and Smart 2001] NA Howgrave-Graham and Nigel P. Smart. 2001. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography* 23, 3 (2001), 283–290.

[Huang et al. 2014] Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C Snoeren, and Kirill Levchenko. 2014. Botcoin: monetizing stolen cycles. In *NDSS '14: Proceedings of the Network and Distributed System Security Symposium*.

[in Bitcoin 2015] Privacy-Enhancing Overlays in Bitcoin. 2015. Sarah Meiklejohn and Claudio Orlandi. In *BITCOIN '15: Proceedings of the 2nd Workshop on Bitcoin Research*.

[Jansen et al. 2014] Rob Jansen, Andrew Miller, Paul Syverson, and Bryan Ford. 2014. From Onions to Shallots: Rewarding Tor Relays with TEARS. In *HotPETs '14: 7th Workshop on Hot Topics in Privacy Enhancing Technologies*.

[Johnson et al. 2013] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. 2013. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *CCS '13: Proceedings of the 20th ACM Conference on Computer and Communications Security*.

[Johnson et al. 2014] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. 2014. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Kaminsky 2011] Dan Kaminsky. 2011. Black Ops of TCP/IP. Presentation. Black Hat USA 2011. http://dankaminsky.com/2011/08/05/bo2k11/

[Karame et al. 2012] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. 2012. Double-spending Fast Payments in Bitcoin. In *CCS '12: Proceedings of the 19th ACM Conference on Computer and Communications Security*. 906–917.

[Kaskaloglu 2014] Kerem Kaskaloglu. 2014. Near Zero Bitcoin Transaction Fees Cannot Last Forever. In *DigitalSec '14: Proceedings of the International Conference on Digital Security and Forensics*. 91–99.

[kiba 2010] kiba. 2010. BitDNS Bounty (3500 BTC). (Dec. 2010). https://bitcointalk.org/index.php?topic=2072.0

[King 2013a] Sunny King. 2013a. PeercoinTalk's Community Interview With Sunny King #1. (Oct. 2013). http://www.peercointalk.org/index.php?topic=2216.0

[King 2013b] Sunny King. 2013b. *Primecoin: Cryptocurrency with prime number proof-of-work*. Technical Report. http://primecoin.io/bin/primecoin-paper.pdf

[King and Nadal 2012] Sunny King and Scott Nadal. 2012. *PPCoin: peer-to-peer crypto-currency with proof-of-stake*. Technical Report. http://peercoin.net/assets/paper/peercoin-paper.pdf

[Klingberg and Manfredi 2002] T. Klingberg and R. Manfredi. 2002. Gnutella 0.6. (June 2002). http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[Koblitz 1987] Neal Koblitz. 1987. Elliptic curve cryptosystems. *Mathematics of computation* 48, 177 (1987), 203–209.

[Kondor et al. 2013] Dániel Kondor, Máton Pósfai, István Csabai, and Gábor Vattay. 2013. Do the rich get richer? An empirical analysis of the BitCoin transaction network. *arXiv preprint arXiv:1308.3892* (2013).

[Koooooj 2013] Koooooj. 2013. Good News for Primecoin: One such coin that exists today that will actually never have ASICs developed for it is Primecoin. (Nov. 2013). http://redd.it/1ra887

[Koshy et al. 2014] Philip Koshy, Diana Koshy, and Patrick McDaniel. 2014. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic. (March 2014).

[Kroll et al. 2013] Joshua A Kroll, Ian C Davey, and Edward W Felten. 2013. The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries. In *WEIS '13: Proceedings of the 12th Workshop on the Economics of Information Security*.

[Kumaresan and Bentov 2014] Ranjit Kumaresan and Iddo Bentov. 2014. How to Use Bitcoin to Incentivize Correct Computations.

In *CCS '14: Proceedings of the 21st ACM Conference on Computer and Communications Security*. 30–41.

[Ladd 2013] Watsonn Ladd. 2013. *Blind Signatures for Bitcoin Transaction Anonymity*. Technical Report. http://wbl.github.io/bitcoinanon.pdf

[Lamport et al. 1982] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

[Larimer 2013] Daniel Larimer. 2013. *Transactions as Proof-of-Stake*. Technical Report.

[Larimer 2014] Daniel Larimer. 2014. *Delegated Proof-of-Stake (DPOS)*. Technical Report. http://v3.bitshares.org/security/delegated-proof-of-stake.php

[Laszka et al. 2015] Aron Laszka, Benjamin Johnson, and Jens Grossklags. 2015. When Bitcoin Mining Pools Run Dry: A Game-Theoretic Analysis of the Long-Term Impact of Attacks Between Mining Pools. In *BITCOIN '15: Proceedings of the 2nd Workshop on Bitcoin Research*.

[Laurie 2011] Ben Laurie. 2011. *Decentralised currencies are probably impossible but letâĂŹs at least make them efficient*. Technical Report. http://www.links.org/files/decentralised-currencies.pdf

[Law et al. 1996] Laurie Law, Susan Sabett, and Jerry Solinas. 1996. How to Make a Mint: The Cryptography of Anonymous Electronic Cash. *American University Law Review* 46, 4 (1996), 1131–1162.

[Lists 2014a] Bitcoin Development Mailing Lists. 2014a. ASIC-proof mining. (2014). http://sourceforge.net/p/bitcoin/mailman/bitcoin-development/thread/53B714A8.1080603@codehalo.com/

[Lists 2014b] Bitcoin Development Mailing Lists. 2014b. Outbound connections rotation. (2014). http://sourceforge.net/p/bitcoin/mailman/bitcoin-development/thread/CAAS2fgRMtgbnokuMYb_MTMo9+uHnpxQ+B9zzd7FcZkE5nLzoHQ@mail.gmail.com/

[Lv et al. 2002] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. 2002. Can heterogeneity make gnutella scalable?. In *IPTPS '02: Proceedings of the 1st International Workshop on Peer-to-Peer Systems*. 94–103.

[Lynch 1989] Nancy Lynch. 1989. A hundred impossibility proofs for distributed computing. In *PODC '89: Proceedings of the 8th ACM Symposium on Principles of Distributed Computing*. 1–28.

[Mackenzie 2010] Adam Mackenzie. 2010. *MEMCOIN2: A Hybrid Proof of Work/Proof of Stake Cryptocurrency with Democratic Control*. Technical Report. http://mc2.xwebnetwork.com/storage/mc2_0.05.pdf

[Malkhi and Reiter 1998] Dahlia Malkhi and Michael Reiter. 1998. Byzantine quorum systems. *Distributed Computing* 11, 4 (1998), 203–213.

[Massias et al. 1999] H. Massias, X. Serret Avila, and J.-J. Quisquater. 1999. Design Of A Secure Timestamping Service With Minimal Trust Requirement. In *the 20th Symposium on Information Theory in the Benelux*.

[Maxwell 2013a] Gregory Maxwell. 2013a. CoinJoin: Bitcoin privacy for the real world. (Aug. 2013). https://bitcointalk.org/index.php?topic=279249.0

[Maxwell 2013b] Gregory Maxwell. 2013b. CoinSwap: Transaction graph disjoint trustless trading. (Oct. 2013). https://bitcointalk.org/index.php?topic=321228.0

[Meiklejohn et al. 2013] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *IMC '13: Proceedings of the 13th ACM SIGCOMM Conference on Internet Measurement*. 127–140.

[Merkle 1987] Ralph C. Merkle. 1987. A Digital Signature Based on a Conventional Encryption Function. In *CRYPTO '87: Proceedings of the 7th Conference on Advances in Cryptology*. 369–378.

[Miers et al. 2013] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. 2013. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *SP '13: Proceedings of the 34th IEEE Symposium on Security and Privacy*. 397–411.

[Miller 2012] Andrew Miller. 2012. Bitcoin Theory (Byzantine Generals and Beyond). (Aug. 2012). https://bitcointalk.org/index.php?topic=99631.0

[Miller et al. 2014] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. 2014. Permacoin: Repurposing Bitcoin Work for Data Preservation. In *SP '14: Proceedings of the 35th IEEE Symposium on Security and Privacy*. 475–490.

[Miller and LaViola Jr 2014] Andrew Miller and Joseph J LaViola Jr. 2014. *Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin*. Technical Report. University of Florida, Computer Science. http://tr.eecs.ucf.edu/id/eprint/78 Working Paper.

[Miller 1985] Victor S Miller. 1985. Use of elliptic curves in cryptography. In *Advances in CryptologyâĂŤCRYPTOâĂŹ85 Proceedings*. 417–426.

[Möller et al. 2003] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. 2003. Mixmaster Protocol — Version 2. IETF Internet Draft. (July 2003).

[Moore and Christin 2013] Tyler Moore and Nicolas Christin. 2013. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In *FC '13: Proceedings of the 17th International Conference on Financial Cryptography and Data Security*. 25–33.

[Möser et al. 2013] Malte Möser, Rainer Böhme, and Dominic Breuker. 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. (Sept. 2013).

[Möser et al. 2014] Malte Möser, Rainer Böhme, and Dominic Breuker. 2014. Towards Risk Scoring of Bitcoin Transactions. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Nakamoto 2008a] Satoshi Nakamoto. 2008a. Bitcoin: A peer-to-peer electronic cash system. (2008).

[Nakamoto 2008b] Satoshi Nakamoto. 2008b. Bitcoin P2P e-cash paper. (Nov. 2008). https://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html

[Nakamoto 2008c] Satoshi Nakamoto. 2008c. Re: Bitcoin P2P e-cash paper. (Nov. 2008). https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html

[Narayanan and Shmatikov 2008] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *SP '08: Proceedings of the 29th IEEE Symposium on Security and Privacy*. 111–125.

[Narayanan and Shmatikov 2009] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing Social Networks. In *SP '09: Proceedings of the 30th IEEE Symposium on Security and Privacy*. 173–187.

[Nielsen 2013] Michael Nielsen. 2013. How the Bitcoin protocol actually works. (Dec. 2013). http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/

[Ober et al. 2013] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. 2013. Structure and Anonymity of the Bitcoin Transaction Graph. *Future Internet* 5, 2 (2013), 237–250.

[O'Dwyer and Malone 2014] Karl J O'Dwyer and David Malone. 2014. Bitcoin Mining and its Energy Footprint. In *ISSC '14: Proceedings of the 25th IET Irish Signals and Systems Conference*.

[Osipkov et al. 2007] Ivan Osipkov, Eugene Y Vasserman, Nicholas Hopper, and Yongdae Kim. 2007. Combating double-spending using cooperative p2p systems. In *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*. 41–41.

[Øverlier and Syverson 2006] Lasse Øverlier and Paul Syverson. 2006. Locating Hidden Servers. In *SP '06: Proceedings of the 27th IEEE Symposium on Security and Privacy*.

[Page et al. 1999] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. (1999).

[Paul et al. 2014] Goutam Paul, Pratik Sarkar, and Sarbajit Mukherjee. 2014. Towards a More Democratic Mining in Bitcoins. In *ICISS '14: Proceedings of the 10th International Conference on Information Systems Security*. 185–203.

[Percival 2009] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions. In *BSDCan '09: The Technical BSD Conference*.

[Pickard et al. 2010] Galen Pickard, Iyad Rahwan, Wei Pan, Manuel Cebrián, Riley Crane, Anmol Madan, and Alex Pentland. 2010. Time critical social mobilization: The darpa network challenge winning strategy. *arXiv preprint arXiv:1008.3172* (2010).

[Plohmann and Gerhards-Padilla 2012] Daniel Plohmann and Elmar Gerhards-Padilla. 2012. Case study of the Miner Botnet. In *CYCON '12: Proceedings of the 4th International Conference on Cyber Conflict*. 1–16.

[Ragan and Salazar 2014] Rob Ragan and Oscar Salazar. 2014. CloudBots: Harvesting Crypto Coins like a Botnet Farmer. Presentation. Black Hat USA 2014. http://www.syscan360.org/slides/2014_EN_CloudBots_RobRaganOscarSalazar.pdf

[Raymond 2000] Jean-François Raymond. 2000. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. 10–29.

[Reed 2014] Stephen L. Reed. 2014. Bitcoin Cooperative Proof-of-Stake. *CoRR: Computing Research Repository* abs/1405.5741 (2014). http://arxiv.org/abs/1405.5741

[Reid and Harrigan 2013] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In *Security and Privacy in Social Networks*. Springer, 197–223.

[Ren 2014] Larry Ren. 2014. *Proof of Stake Velocity: Building the Social Currency of the Digital Age*. Technical Report. http://www.reddcoin.com/papers/PoSV.pdf

[Rivest et al. 2001] Ronald L Rivest, Adi Shamir, and Yael Tauman. 2001. How to leak a secret. 552–565.

[Ron and Shamir 2013] Dorit Ron and Adi Shamir. 2013. Quantitative Analysis of the Full Bitcoin Transaction Graph. In *FC '13: Proceedings of the 17th International Conference on Financial Cryptography and Data Security*. 6–24.

[Ron and Shamir 2014] Dorit Ron and Adi Shamir. 2014. How Did Dread Pirate Roberts Acquire and Protect His Bitcoin Wealth?. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Rosenfeld 2011] Meni Rosenfeld. 2011. Analysis of Bitcoin Pooled Mining Reward Systems. *CoRR: Computing Research Repository* abs/1112.4980 (2011). http://arxiv.org/abs/1112.4980

[Rosenfeld 2012a] Meni Rosenfeld. 2012a. Analysis of hashrate-based double spending. (2012). https://bitcoil.co.il/Doublespend.pdf

[Rosenfeld 2012b] Meni Rosenfeld. 2012b. Overview of colored coins. (2012). https://bitcoil.co.il/BitcoinX.pdf

[Ruffing et al. 2014] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2014. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. In *HotPETs '14: 7th Workshop on Hot Topics in Privacy Enhancing Technologies*.

[S. 2014] Michael S. 2014. *Why CoinJoin, as Used in DarkCoin,does NOT bring Full Anonymity*. Technical Report. http://www.scribd.com/doc/227369807/Bitcoin-Coinjoin-Not-Anonymous-v01

[Saxena et al. 2014] Amitabh Saxena, Janardan Misra, and Aritra Dhar. 2014. Increasing Anonymity in Bitcoin. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Schwartz 2011] Aaron Schwartz. 2011. Squaring the Triangle: Secure, Decentralized, Human-Readable Names. (Jan. 2011). http://www.aaronsw.com/weblog/squarezooko

[Schwartz et al. 2014] David Schwartz, Noah Youngs, and Arthur Britto. 2014. *The Ripple Protocol Consensus Algorithm*. Technical Report. https://ripple.com/files/ripple_consensus_whitepaper.pdf

[SECG 2000] SECG. 2000. SEC 2: Recommended Elliptic Curve Domain Parameters. (2000). http://www.secg.org/collateral/sec2_final.pdf

[Serjantov et al. 2002] Andrei Serjantov, Roger Dingledine, and Paul Syverson. 2002. From a Trickle to a Flood: Active Attacks on Several Mix Types. In *IHW '02: Proceedings of the 5th International Workshop on Information Hiding*.

[Shamir 1979] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[Sompolinsky and Zohar 2013] Yonatan Sompolinsky and Aviv Zohar. 2013. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *IACR: Cryptology ePrint Archive* 2013 (2013). Issue 881.

[Spagnuolo et al. 2014] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. BitIodine: Extracting Intelligence from the Bitcoin Network. In *FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security*.

[Syverson et al. 2000] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. 2000. Towards an Analysis of Onion Routing Security. In *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. 96–114.

[Szabo 1997] Nick Szabo. 1997. The Idea of Smart Contracts. (1997).

[Szabo 1998] Nick Szabo. 1998. Secure Property Titles with Owner Authority. (1998).

[Szabo 2003] Nick Szabo. 2003. Advances in Distributed Security. (2003).

[Szabo 2005] Nick Szabo. 2005. Bit Gold. (2005). http://unenumerated.blogspot.de/2005/12/bit-gold.html

[Szefer and Lee 2013] Jakub Szefer and Ruby B. Lee. 2013. BitDeposit: Deterring Attacks and Abuses of Cloud Computing Services through Economic Measures. In *CCGRID '13: Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. 630–635.

[Turek and Shasha 1992] John Turek and Dennis Shasha. 1992. The many faces of consensus in distributed systems. *IEEE Computer* 25, 6 (1992), 8–17.

[van Saberhagen 2013] Nicolas van Saberhagen. 2013. *CryptoNote v2.0*. Technical Report. https://cryptonote.org/whitepaper.pdf

[Vandervort 2014] David Vandervort. 2014. Challenges and Opportunities Associated with a Bitcoin-Based Transaction Rating System. In *FC '14: Proceedings of the 18th International Conference on Financial Cryptography and Data Security*. 33–42.

[Vasek and Moore 2015] Marie Vasek and Tyler Moore. 2015. There's No Free Lunch, Even Using Bitcoin: Tracking the Popularity and Profits of Virtual Currency Scams. In *FC '15: Proceedings of the 19th International Conference on Financial Cryptography and Data Security*.

[Vasek et al. 2014] Marie Vasek, Micah Thornton, and Tyler Moore. 2014. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *BITCOIN '14: Proceedings of the 1st Workshop on Bitcoin Research*.

[Vector67 2011] Vector67. 2011. Fake Bitcoins? (2011). https://bitcointalk.org/index.php?topic=36788.msg463391#msg463391

[vinced 2011] vinced. 2011. Namecoin - a distributed naming system based on Bitcoin. (April 2011). https://bitcointalk.org/index.php?topic=6017.0

[Vornberger 2012] Jan Vornberger. 2012. Marker addresses: Adding identification information to Bitcoin transactions to leverage existing trust relationships. In *INFORMATIK '12: Proceedings of the 42nd GI Jahrestagung*. 28–38.

[Warren 2012] Jonathan Warren. 2012. Bitmessage: A peer-to-peer message authentication and delivery system. (Nov. 2012). https://bitmessage.org/bitmessage.pdf

[Wilcox-O'Hearn 2010] Zooko Wilcox-O'Hearn. 2010. Names: Distributed, Secure, Human-Readable: Choose Two. (Oct. 2010). http://web.archive.org/web/20011020191610/http://zooko.com/distnames.html

[Wilkinson et al. 2014] Shawn Wilkinson, Jim Lowry, and Tome Boshevski. 2014. *MetaDisk A Blockchain-Based Decentralized File Storage Application*. Technical Report. http://metadisk.org/metadisk.pdf

[Willett 2013] J. R. Willett. 2013. The Second Bitcoin Whitepaper. (2013). https://sites.google.com/site/2ndbtcwpaper/2ndBitcoinWhitepaper.pdf

[Wood 2014] Gavin Wood. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. (2014). http://gavwood.com/Paper.pdf

[Wuille 2014] Pieter Wuille. 2014. BIP 62: Dealing with malleability. (March 2014). https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki

[Zimmermann 1995] Philip R. Zimmermann. 1995. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA.