Agreement with Satoshi – On the Formalization of Nakamoto Consensus

Nicholas Stifter*†, Aljosha Judmayer*, Philipp Schindler*, Alexei Zamyatin‡*, Edgar Weippl*†
*SBA Research, ‡Imperial College London,
†Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI), TU Wien
Email: (firstletterfirstname)(lastname)@sba-research.org

***Abstract*—The term *Nakamoto consensus*[1] is generally used to refer to Bitcoin's novel consensus mechanism, by which agreement on its underlying transaction ledger is reached. It is argued that this agreement protocol represents the core innovation behind Bitcoin, because it promises to facilitate the *decentralization* of trusted third parties. Specifically, Nakamoto consensus seeks to enable mutually distrusting entities with weak pseudonymous identities to reach eventual agreement while the set of participants may change over time. When the Bitcoin white paper was published in late 2008, it lacked a formal analysis of the protocol and the guarantees it claimed to provide. It would take the scientific community several years before first steps towards such a formalization of the Bitcoin protocol and Nakamoto consensus were presented. However, since then the number of works addressing this topic has grown substantially, providing many new and valuable insights. Herein, we present a coherent picture of advancements towards the formalization of Nakamoto consensus, as well as a contextualization in respect to previous research on the agreement problem and fault tolerant distributed computing. Thereby, we outline how Bitcoin's consensus mechanism sets itself apart from previous approaches and where it can provide new impulses and directions to the scientific community. Understanding the core properties and characteristics of Nakamoto consensus is of key importance, not only for assessing the security and reliability of various *blockchain* systems that are based on the fundamentals of this scheme, but also for designing future systems that aim to fulfill comparable goals.**

I. The Importance of Nakamoto Consensus

The recent explosive increase in both economic valuation and technical interest towards Bitcoin, cryptocurrencies, and distributed ledgers in general, is mirrored by equally growing research efforts from the scientific community to better understand, employ and extend upon the fundamental principles that govern these technologies. Not only has the body of peer-reviewed literature directly related to Bitcoin and cryptocurrencies increased substantially as outlined by this recent taxonomy [XWS+17], but a lot of work is also presented online in the form of pre-prints, e-prints, and informally as blog posts, following the publishing spirit of the original Bitcoin white paper [Nak08a].

Moreover, many other research fields are also exploring how the underlying concepts behind blockchain technologies could be applied in their domain. This renders it difficult for both researchers and practitioners to get a coherent picture of the state-of-the-art in this emerging field. We therefore believe that further systematization efforts related to Bitcoin and blockchain technologies, following the comprehensive overview of research perspectives and challenges for Bitcoin presented by Bonneau et al. in 2015 [BMC+15], are necessary.

In particular, the study and formalization of the Bitcoin protocol and its underlying Nakamoto consensus has seen significant advances in recent years (e.g. [KP15], [GKL16], [BPS16a]) that are not yet systematically exposed. Recent work provides a broad overview of different consensus mechanisms in the context of blockchain technologies [BSAB+17], however we feel that a more in-depth analysis of the relationship between Nakamoto consensus and previous approaches to Byzantine consensus is still outstanding. We hereby narrow this gap by relating research towards Nakamoto consensus to other key insights and aspects on the topic of consensus.

Consensus is a fundamental building block in fault tolerant and distributed computing, and the guarantees a consensus protocol provides can greatly impact the overall security and reliability of (distributed ledger) systems [CV17]. Bitcoin promises to solve the *double spending* problem in a distributed, peer-to-peer environment without the necessity to rely on a trusted third party by enabling participants to reach (eventual) *agreement* on the state changes to a shared transaction ledger. Nakamoto consensus hence lies at the core of this system.

Without an in-depth understanding of this mechanism, entire categories of newly designed systems, as well as the applications that are built on top of them, are potentially vulnerable to attacks [NG16], [CV17]. Modifications to consensus related rules, even if they appear small or straightforward, can fundamentally impact underlying incentives and greatly affect security guarantees [ZP17]. As the ecosystem around "blockchain" has grown into a multi-billion dollar industry, severe failures could have far-reaching consequences and a long-term negative impact on the entire field.

On the other hand, fundamental insights on Nakamoto consensus have already spawned novel and hybrid consensus approaches that exhibit interesting properties and characteristics, while providing the necessary frameworks to analyze and evaluate the correctness and security of such approaches. Combining aspects of "classical" Byzantine fault tolerant (BFT) consensus protocols with Nakamoto consensus may help to address increasing concerns regarding the scalability and performance of blockchain technologies.

Motivated by this emerging new area of research, we set out to paint a coherent picture of the insights and findings that have been presented on the topic of Nakamoto consensus and the fundamental mechanisms behind Bitcoin and similar blockchain protocols.

[1]One of the earliest uses of the term *Nakamoto consensus* can be attributed to a blog post by Nick Szabo in [Sza14], after which it appeared in scientific publications such as [BMC+15], [LTKS15].

II. The Consensus Problem

"It is not much matter which we say, but mind, we must all say *the same*."
– Lord Melbourne, [Bag00]

The *agreement* or *consensus problem* is a long standing research topic that has, in particular, been the subject of much discussion in the field of fault tolerant and distributed computing [Fis83]. Consensus lies at the core of many distributed algorithms and is one of the most fundamental problems in this field [FLP85]. It can, for instance, serve as a basis for achieving active replication, such as in the *replicated state machine* approach [Sch90], or be used to implement any *wait-free* concurrent data object among a set of processes [Her91]. The term *distributed* in this context does not necessitate large geo-spatial differences and can also refer to processors within a single system that communicate via message passing or shared memory [Lyn96].

To effectively outline how Nakamoto consensus relates to the consensus problem domain, we first give a brief overview of the history and some of the key insights that have been presented in this field before the advent of blockchain technologies. The presentation of definitions is deferred, where possible, to the next Subsection (II-B), which places its focus on consensus problem definitions and outlines how they, as well as the assumed system model, impact the solvability of consensus.

A. Brief History of the Consensus Problem

One of the earliest publications associated directly with the consensus problem was presented in the beginning of the 1980s by Pease et al. on the topic of *interactive consistency* [PSL80]. Shortly thereafter, the same set of authors went on to publish their seminal work on the *Byzantine generals problem* [LSP82], which is closely related to interactive consistency. Both works deal with the question of how a (fixed) group of participants can reach agreement upon a value or set of values, if participants are allowed to deviate *arbitrarily* from the prescribed protocol. Under the assumption of a relatively strong system model, they were able to show that strictly less than a third of the participants are allowed to exhibit such arbitrary behavior, if the defined properties of the problem specification are to hold. These works played a key role in stepping loose an entirely new field of research, centered around formalizing and characterizing various consensus problems [Fis83].

Another seminal work on the topic of consensus is Fischer et al. [FLP85], which is referred to as the *FLP impossibility result*. It shows that *deterministic* consensus becomes impossible in a completely *asynchronous* system, even if only a single process is allowed to fail in the *crash-stop* model and communication between processes is reliable. The FLP impossibility result inspired research on the minimal synchrony assumptions necessary to be able to reach consensus, leading to the definition of different models of *partial synchrony* [DDS87], [DLS88].

Instead of strengthening the system model and its assumptions, the impossibility result can also be circumvented by relaxing the problem definition, such as only requiring *probabilistic* guarantees for aspects, such as correctness or termination of the algorithm. In particular, the class of so called *randomized* consensus algorithms, pioneered by Ben-Or [BO83] and Rabin [Rab83], has received particular attention.

Nevertheless, at the time, the take-away from these results was that systems for reaching consensus in the presence of arbitrary, or so called *Byzantine failures*, while in principle feasible, were largely impractical for real-world scenarios [CL02] due to the overhead incurred in additional communication and computation complexity.

It would take over a decade until publications such as *"Practical Byzantine Fault Tolerance"* (PBFT) from Castro and Liskov [CL$^+$99] showed that so called *Byzantine fault tolerant* (BFT) consensus algorithms could indeed be practicable under realistic system assumptions. Nevertheless, while research on the topic of BFT consensus was ongoing [CKS00], [CWA$^+$09], [GKQV10], [VCB$^+$13], it remained a comparatively isolated topic area, given the broad range of potential applications. In part, this may be attributed to the fact that consensus protocols are often discussed in the context of *state machine replication* [Lam84], [Sch90] and achieving active replication for services, such as databases. Here, all replicas, i.e participants, may be under the control of a single entity and the more benign crash-fault tolerance can be a tenable system model. In particular, Lamport's crash-fault tolerant *Paxos* consensus algorithm [Lam98] and derivations thereof have found their way into practical applications [CGR07]. Another important contribution is the concept of using *failure detector* abstractions as oracles, instead of relying on concrete timing assumptions in the design of consensus protocols [CT96]. Different classes of failure detector oracles are formed, that open up the ability to formally define the minimal guarantees they need to provide to be able to solve a particular consensus problem.

Consensus protocols are often considered and formally analyzed assuming a *static* set of participants. In practice, however, it is a desirable property to be able to dynamically reconfigure this set. Reaching agreement on a dynamically changing set of processes is, in itself, a problem related to consensus, namely the *Group Membership Problem* [Cri91], [Ric93]. It has been studied primarily in the crash-failure model, where the introduction of the concept of *virtual synchrony* by Birman and Joseph [BJ87] in the *ISIS* system has been influential and led to a variety of practical group membership systems [Ban98], [AS98], [MPR01].

So far, the presented works primarily assume a system model, where processes communicate with each other through means of *message passing* over some channel. Agreement problems have also been studied in alternative models, such as *shared memory*, where consensus was shown to be universal [Her91], [MMRT03]. Depending on the system assumptions, the FLP impossibility result also applies to shared memory [Her88]. In the context of Nakamoto consensus, the primary interest lies in the former message passing model, as it more closely resembles the actual system.

B. Defining Agreement Problems

We depict the formalization of consensus problems, loosely following the general style of work in the field of distributed algorithms outlined by Lynch [Lyn96], as three distinct steps:

I) Identification and abstraction of (practical) problems of significance by characterization through a set of properties and desired guarantees.
II) Assumption of an underlying (mathematical) system model of the distributed system, in which the problem is to be solved.
III) Precise specification and proof that the developed algorithms solve I) in II); Analysis of the algorithms' complexities and proofs of bounds and limitations or their impossibility.

We outline aspects of these three steps in more detail to provide a basis for comparison and discussion towards the formalization efforts on Nakamoto consensus, beginning with Step II), i.e. the system model, as it offers a good opportunity to introduce a common terminology.

1) System Models for Consensus Problems: Assuming a system model that provides very weak guarantees may render a (consensus) problem very hard or impossible to solve. On the other hand, while overly strong guarantees might allow for an easy solution, achieving these guarantees in itself can turn into a hard problem.

Processes and Connectivity. For consensus problems, the distributed system is commonly modeled as a static, bounded number of processes $\{p_1, p_2, \ldots, p_n\} = \Pi$, where communication between processes occurs by *message passing* over *reliable point-to-point links*. A *process*[2] is an abstract unit able to perform computations in a distributed system [CGR11]. The term *correct* is used only if during the entirety of an execution the particular component, such as a process or communication link, will exhibit no faulty behavior or deviate from the prescribed protocol rules.

The communication graph is generally assumed to be *bidirectional* and *completely connected*, however, other topologies have also been considered [Dol81], [LSP82]. In [LSP82] a distinction is made between *oral* and *authenticated*[3] *messages* and they are defined by the following characteristics:

Definition 1: Oral messages:

1) Every message that is sent is delivered correctly.
2) The receiver of a message knows who sent it.
3) The absence of a message can be detected.

In the context of the system model assumed in [Lam84], the second property, in particular, is needed or else a single malicious process could defeat any distributed consensus algorithm. Authenticated messages extend oral messages by the following fourth property:

Definition 2: Authenticated messages:

4) a) Messages sent by a correct process cannot be forged, and any alteration of the contents of these signed messages can be detected.
   b) Anyone can verify the authenticity of a correct process's signature.

Static and Dynamic Distributed Systems. Models where both the set of processes and their communication links remain static are referred to as *static distributed systems*. Interestingly, there are no widely agreed upon definitions for *dynamic system model* counterparts. In [BBRTP07] Baldoni et al. investigate two attributes which they consider defining characteristics of dynamic distributed systems, namely the varying size of the system over time and its topology in terms of the process neighborhood.

Unless otherwise specified, we assume a *static system model* with a fully connected communication graph of reliable point-to-point links when describing consensus protocols.

Synchrony Assumptions. One essential property of the system model that greatly influences the solvability of consensus are its *synchrony assumptions*. In their seminal work, Fischer, Lynch and Patterson show that reaching *deterministic* agreement in a system with asynchronous communication is impossible, even if message communication is reliable and only a single process can fail (in the crash-stop model) [FLP85].

Effectively, without bounded delays on message transmission times, it is impossible to deterministically decide whether a process has failed or its messages have simply not yet arrived. To ensure that the *Agreement* property of consensus under such conditions cannot be violated, the *Termination* property is no longer satisfiable, as a single failed process could require all correct processes to wait indefinitely for an answer. This fundamental insight, which is commonly referred to as the *FLP impossibility result*, outlines an important limitation of all problems in the consensus domain. For practical real-world systems, simply assuming stronger synchrony does not fully address this issue, because components have a non-zero probability of failure, and hence also render such synchrony assumptions at best probabilistic. Therefore, it is necessary to contemplate the possibility of timing failures and choose a suitable trade-off between availability and correctness, because no protocol can exist that guarantees both properties at all times.

Failures and Failure Detection. To be able to reason about failures it is first necessary to define how the processes and communication links that make up the system can actually fail in a particular system model. A protocol is considered to be *f-resilient* if it tolerates no more than f faulty processes of the n processes that make up the system. The following list is a (non-exhaustive) generalization of failure types that can be considered:

- *Crash failure.* A basic failure model where components are assumed to crash and never recover.
- *Omission failure.* Here components may omit actions such as sending messages or performing computations. Assuming the ability for *crash recovery* also falls into this category.
- *Timing failure.* Timing failures occur when synchrony assumptions are violated. In an asynchronous system, this

[2]Originally referred to as processors [PSL80].

[3]In [LSP82], Lamport et al. called them signed *written* messages.

failure is irrelevant.

- *Byzantine failure.* Byzantine failures (sometimes also referred to as *arbitrary failures*) allow a component to deviate arbitrarily, and hence also maliciously, from its expected behavior. This includes duplicating or changing message contents, sending unsolicited messages, and temporarily or permanently exhibiting any of the previously listed failure characteristics.

As previously outlined in Subsection II-A, Fischer et al. show the impossibility for deterministic consensus in a fully asynchronous system model [FLP85] because processes cannot positively decide if another process has failed.

Chandra and Toueg [CT96] introduces the concept of failure detectors as a form of *oracle*, which a process can query. Here, instead of considering concrete synchrony assumptions, problems rely on a distributed failure detector abstraction, based on local unreliable failure detector modules to determine whether a process has failed. These abstract failure detectors are characterized by properties such as *completeness* and *accuracy*, while concrete timing assumptions and requirements are shifted towards implementations that provide the defined properties. Consensus problems can hence purely rely on, and be analyzed and classified through, the abstract class of failure detectors required to solve them. Chandra and Toueg define eight different classes of failure detectors based on two completeness and four accuracy properties and show that the weakest class of failure detectors required to solve consensus with crash failures in asynchronous systems lies in the class of $\diamond\mathcal{W}$[4].

Doudou et al. [DGG02] outline that the detection of Byzantine behavior of a process p by a Byzantine failure detector cannot be entirely independent of the algorithm $\mathcal{A}$ in which the failure detector is used. Kihlstrom et al. [KMMS03] also point out that there is a subset of Byzantine faults that cannot be detected.

In [MR97], Malkhi and Reiter use an approach where a Byzantine failure detector class $\diamond\mathcal{S}(bz)$ is defined, which only detects (quiet) behaviors that may prevent progress and defers all other forms of Byzantine failure detection to upper levels of the consensus protocol. Failure detector checking if a process stops sending messages have also been defined by Doudou et al. as so-called *muteness failure detectors*, denoted by $\diamond M_{\mathcal{A}}$ [DGG02].

Hybrid System Models. *Wormholes*, introduced in Verissimo [Ver03], are closely related to the notion of *architectural hybridization* and encapsulate and provide stronger guarantees to an otherwise weaker environment [CVNV11]. Instead of presenting an abstraction that specifies the minimum requirements (such as the ability to detect failures), wormholes provide the ability to introduce controllable levels of predictability into systems that are otherwise mostly uncertain with regard to their provided guarantees.

By relying on wormholes, it is possible to further improve upon bounds, such as the resilience to failures, which would otherwise not be possible. Correia et al. [CVL10], for instance, shows how wormholes can be used to transform any *indulgent* consensus algorithm that tolerates crash failures with $n \geq 2f+1$ processes into one that tolerates Byzantine failures with $n \geq 2f+1$ processes, even if the system model is assumed to be asynchronous or partially synchronous.

This result does not contradict previously established lower bounds for Byzantine consensus in such models, which is $n \geq 3f+1$, because the system is in fact *hybrid*, where the stronger (synchronous) system model of the wormholes renders these results possible[5].

2) Consensus Problem Definitions: Consensus problems and variations thereof are generally characterized through a set of properties similar or equal to Definition (3). The *Validity* and *Agreement* properties are referred to as *safety* properties because they guard against trivial solutions or solutions violating the desired consensus assumptions, while the *Termination* property ensures *liveness*, i.e., that the algorithm eventually makes progress and produces some result [Fuz08].

Definition 3: The *agreement-* or *consensus problem*:

- *Validity:* If a process decides a value v, then v was proposed by some process.
- *Agreement:* No two correct processes decide differently.
- *Termination:* Every correct process (eventually) decides some value.

In Definition 3 the *validity* property only ensures the decision value was the vote of any process partaking in consensus. A *decision* is reached once a process irreversibly chooses a value. However, this constraint in itself does not guarantee that any meaningful value will be chosen. In [Nei94] the concept of *strong validity* is presented, where the particular value agreed upon must also have been proposed by a *correct* participant. The property hence is as follows:

- *Strong Validity:* If a process decides a value v, then v was proposed by some correct process

It is shown that solving strong validity in a Byzantine failure setting requires a trust assumption of $n > max(mf, 3f)$, where m is the size of the set from which input values to consensus are chosen, $m = |V|$. Clearly, strong validity can greatly increase the necessity for a large proportion of correct processes, depending on the size of the permissible input set of values the processes may propose. Other properties, such as *median validity* [SW16], may require characteristics of the proposals such as being *orderable* to provide more meaningful guarantees on decision values than the initially presented validity property, while incurring less overhead than strong validity.

Consensus protocols terminate when all correct processes have halted. If this is achieved in the same communication round, the processes are considered to have reached *immediate agreement*, otherwise they reach *eventual agreement* [Fis83]. For *randomized consensus*, the *termination* property of the *consensus problem* is weakened to:

Definition 4: Termination property for *randomized consensus*:

[4]More accurately, they show that the weakest class of failure detector is $\diamond\mathcal{S}$, and present an algorithm that transforms failure detectors of class $\diamond\mathcal{W}$ into $\diamond\mathcal{S}$

[5]In a synchronous system with authenticated messages, Byzantine consensus is possible for $n \geq f \geq 0$ [PSL80].

- *Termination with Probability 1:* Every correct process eventually decides some value with probability 1.

What this means is that, rather than requiring all permissible executions of a protocol to eventually terminate, executions in the randomized approach may not actually terminate, but this occurs with probability 0 as the number of communication rounds R approaches ∞.

Some randomized consensus solutions may also consider a model where instead of *termination*, the *agreement* property is weakened, such that consensus is always reached within a *finite* number of rounds, albeit only with probability $1-\alpha$, and a probability α of error. Such protocols are sometimes referred to as *Monte Carlo* randomized consensus algorithms [IT08]. In [Rab83], for example, Rabin presents a protocol where, for a fixed number of rounds R, the probability of error is $\alpha = 2^{-R}$. Randomized consensus with a non-deterministic agreement property has been largely overlooked as a topic of research [CVNV11].

Different agreement problems. There exist various problems that are either variants of, or are closely related to the consensus problem, such as:

- *Binary consensus*, which represents the most reduced form of consensus, where processes only need to agree on a single binary digit, i.e., the value $v \in V$ that a process can select as their proposal is in the set $V = \{0, 1\}$. Binary consensus is often encountered when formally describing or modeling consensus protocols and their properties [FL82], [BO83], [FLP85] and can be transformed to *multivalued* consensus where the set of possible proposal values V can be arbitrarily large [MRT00].
- Reaching agreement on a vector of values, either called *vector consensus* [CVNV11] or referred to as *interactive consistency*, if agreement should also be reached on whether participating processes are faulty. The desired goal is to ensure that a (fixed) set of processes of size n reaches agreement upon the same vector of values $\{v_1, v_2, \ldots, v_n\} = V$, where v_i generally corresponds to some private value of process p_i, and where a subset of at most f processes, $f \leq n$, may fail arbitrarily. Based on the assumed system model, it was shown that an arbitrary[6] number of failures can be tolerated, i.e., $n \geq f \geq 0$, if *authenticated messages* are available. On the other hand, under the assumption that only weaker *oral messages* can be used, Pease et al. show that at least $n \geq 3f + 1$ participants are required to tolerate f faulty processes [PSL80].
- *Total Order Broadcast* (also referred to as *atomic broadcast*) [DSU04], where messages sent to a set of processes are to be delivered by processes in the same *total order*.
- *State Machine Replication* (SMR) [Sch90], where agreement is generally to be reached on both the input and its ordering to a set of replicated deterministic state machines, such that all replicas receive and process the same sequence of requests. It is clear that atomic broadcast and SMR are intimately related, and the latter can be readily implemented by building on top of the former. However, depending on the exact problem definition, ordering constraints for the inputs may be relaxed for SMR as long as agreement on the state machines' states is ensured.
- *The Group Membership Problem* (GMP) [Ric93], [Cri91], where agreement by a set of processes on whether they belong to a particular group is to be reached; thereby additional processes may *join*, while existing or failed processes may be removed from or *leave* that group.
- *Terminating Reliable Broadcast* (TRB) [HT94], where a distinguished *sender* from a set of processes is to disseminate a message to this set, so that all *correct processes* either agree upon the receipt of the message or that the sender is faulty.
- *The Byzantine Generals Problem* which requires the subset of non-faulty processes to reach consensus either upon the private value, i.e. message, of a *predetermined* leader, or that this leader is faulty. Therein, assuming the same system model as [PSL80], Lamport et al. show that the failure bounds of $f < \lceil \frac{n}{3} \rceil$ for oral messages, and $f \leq n$ for authenticated messages, also apply to this problem [LSP82].
  A priori agreement by all participants on the leader, i.e. the sender of the message, renders the Byzantine generals problem a *broadcast protocol*. In particular in the context of the Byzantine generals problem the terminology over the years is inconsistent and *Byzantine agreement* (BA) is sometimes used to refer to consensus in a Byzantine failure model *or* the Byzantine generals problem. To avoid confusion we exclusively associate BA with byzantine consensus.

Some of these agreement problems have been shown to be equivalent to consensus, such as total-order broadcast, while others, such as TRB, may be harder [CT96].

III. Agreement Problems related to Nakamoto Consensus

This section establishes connections between previous research on consensus problems and some of the characteristic properties and goals attributed to Bitcoin and Nakamoto consensus. A general overview of both Bitcoin and current research perspectives and challenges in the field is given by publications such as [BMC+15], [TS16], [Nar16], [YHKC+16].

The Double Spending Problem. To prevent users from spending the same virtual currency more than once, generally referred to as *double spending*, some form of *agreement* needs to be reached on the ordering and state of transactions in the system. Previous proposals for cryptographic currencies either required some form of *trusted third party* [CFN90], [Fin04] or made unrealistic system assumptions [Dei] to deal with the double spending problem.

Nakamoto consensus is often related to the concept of *state machine replication* and it has been shown that the latter can satisfy the desirable properties of a distributed public ledger [BPS16a]. To the best of our knowledge there has not been a formal analysis of how the problem of preventing double spending or achieving a public transaction ledger actually

[6]However, the problem becomes vacuous if $n < f + 2$ [LSP82].

relates to other agreement problems, and if these problems are equivalent or weaker than consensus.

In [Hoe07] Hoepman presents a solution to the distributed double spending problem for peer-to-peer networks. This is achieved by employing both deterministic and random *clerk sets* of participants for validation, where these sets follow certain size constraints to reduce the communication overhead for large-scale systems. The assumed system model is static, where f of the n participants are assumed to act maliciously. It has an asynchronous fully connected reliable network, relies on a Public-Key-Infrastructure (PKI), and coin transmission is an atomic process, i.e., coins can not be spent concurrently.

It is shown that double spending can be deterministically prevented with fixed clerk sets of size $2\sqrt{n(f+1)}$, and different bounds for random clerk sets are given such that double spending is prevented with overwhelming probability.

Consensus Finality. Brewer suggests that there is an inherent trade-off between *consistency*, *availability*, and *partition tolerance* in distributed systems, where only two of the three properties can be achieved in practice [Bre00]. Gilbert and Lynch have more formally considered this conjecture and shown an impossibility result for the *asynchronous* network model, and present solutions for the *partially synchronous model* [GL02]. In practice synchrony assumptions are at best probabilistic [CT96], in particular if we consider a peer-to-peer protocol over the Internet. Termination or agreement properties can be weakened to hold only probabilistically in order to deal with asynchrony, as we have outlined in Section II. Bitcoin does not provide deterministic guarantees for *agreement*, and the protocol in itself is not designed to terminate after some finite amount of steps, in particular trading *consistency* for *partition tolerance*. State changes commited to the transaction ledger are rendered probabilistic, and a *decision* on a particular state change only reaches $Pr(1)$ as the number of rounds r approaches $\lim_{r\to\infty}$.

This aspect of non-deterministic agreement in Nakamoto consensus has led to the term *consensus finality* to characterize and differentiate between consensus protocols that *decide* with certainty and those where a decision only stabilizes eventually [Vuk16].

In *Stabilizing consensus* agreement is eventually reached, however, while the execution proceeds decision values of processes may changes finitely often until ,after some point, the system reaches a stable configuration [AFJ06]. The *k-agreement problem* [BT16] weakens the agreement problem to only require that all decisions are in a set of k values. Another weakened variation is the *approximate agreement problem* [DLP$^+$86] which allows inputs, decisions, and messages to be real numbers and requires the difference between any two decision values to be within a small tolerance and that any decision value is within the range of input values.

Dynamic Membership with Byzantine Faults. The so far presented ”classical” BFT protocols assume a static set of known processes that make up the consensus participants. Byzantine fault tolerance in the context of (dynamic) group membership systems is considered in systems such as *Rampart* [Rei96] and *SecureRing* [KMMS01]. However, in these cases either the problem of dealing with potentially Byzantine processes in the changing membership set is only partially addressed (Rampart), or relatively strong guarantees on aspects such as *synchrony* and the ability to detect Byzantine failures (SecureRing) are assumed.

In particular, assuming that an adversary can generate multiple identities renders consensus difficult under a Byzantine failure model. Such *Sybil* attacks [LSM06] were first outlined by Douceur [Dou02], and generally demand strong system assumptions that seem unrealistic to achieve in decentralized, peer-to-peer environments. In [AJK05] Aspnes et al. show that *moderately hard puzzles* [DN92], [RSW96], [JB99] can be employed to establish priced identities, such that both honest and Byzantine participants only generate identities proportional to their respective computational power in order to address the Sybil attack. It is argued that after an initial collection phase these identities can be used for standard authenticated Byzantine agreement protocols. However, the model assumes a static set of participants and synchronous and reliable communication and does not consider a dynamic system model where processes may join or leave the system.

A model where ”anyone”, i.e., *pseudonymous* or *anonymous* entities, can (in principle) participate in the consensus process is sometimes referred to as *permissionless*; systems that rely on some predetermined set of consensus processes are called *permissioned* [Swa15], [Vuk15]. We note, however, that these terms have not yet been precisely defined and may vary in their exact meaning.

Unknown and Anonymous Participants.

Consensus problems have also been studied in *anonymous* networks and under the assumption that the set of participants is previously unknown.

Alchieri et al. [ABSFG08] considers Byzantine fault tolerant consensus with unknown participants (BFT-CUP) and presents a solution that does not require digital signatures. It is assumed that the Sybil attack is infeasible and communication is based on authenticated and reliable point to point channels between known processes.

In [BR09] different classes of *anonymous* failure detectors are introduced for crash-fault tolerant consensus in the anonymous setting. Delporte et al. consider Byzantine agreement in a static system of n participants where processes can have homonyms, i.e. there is a set of id's $l < n$ where processes share the same id [DGFGK10]. They show that in a such system, Byzantine consensus is only possible if for the set of all id's l it holds that $l > 3f$.

Consensus Scalability. Nakamoto consensus can potentially support a large number of processes that concurrently participate in the consensus process. In [Vuk15], Vukolić outlines the different properties both the ”classical” BFT consensus approaches and Nakamoto consensus provide when applied to the context of distributed ledgers.

The high redundancy requirements for (Byzantine) consensus have generally led to the notion that such systems are impractical for real-world scenarios [CNV04]. While Castro et al. were able to show the practicability for byzantine fault tolerance [CL$^+$99], the problem still remains that the system must be comprised of $n \geq 3f+1$ participants. Hybrid system approaches using *wormholes* manage to reduce this to $n \geq 2f + 1$ at the cost of requiring stronger synchrony assumptions for the wormhole [CVL10], [CNV04].

A problem many classical BFT consensus systems face is the difficulty of efficiently scaling with respect to the number of processes that can actively participate in consensus. The message complexity is usually expected to be quadratic, i.e., $O(n^2)$, and historically BFT systems generally assume a relatively small number of processes, ranging in the tens to at most hundreds of participants. Beyond the communication overhead for a large set of processes, the focus on small consensus groups may also be attributed to the fact that BFT protocols were often developed in the context of state machine replication in order to provide fault-tolerant replication to some particular service, such as a database. Solutions for supporting a larger set of processes in such models may involve the delegation of consensus responsibilities to a select subset of nodes that are responsible for collecting local peer information and including it in their consensus votes, as well as disseminating consensus results.

Player Incentives. Considering that a subset or all consensus participants act *rationally* instead of taking on the classical honest or *correct* and dishonest (i.e. *Byzantine*) roles, is still largely an open research question. In [AAC$^+$05] Aiyer et al. introduce the BAR (Byzantine, Altruistic, Rational) model as a foundation for reasoning about cooperative services. Li et al. [LCW$^+$06] present a peer-to-peer data streaming application that builds upon this BAR model and presents a BAR-tolerant *gossip protocol*. In [GKTZ12] Groce et al. considers Byzantine agreement and broadcast protocols under the assumption that a subset of the consensus participants are *rational* adversaries and there exists either partial or complete knowledge of the adversary's preferences. Based on this they are able to show that many of the known impossibility results and bounds in the traditional model do not hold in the rational model.

IV. Formalizing Nakamoto consensus

When the Bitcoin white paper was first presented in late 2008, it was met by many researchers with skepticism, as the paper lacked a formal analysis of the protocol and its claimed guarantees [BMC$^+$15]. In respect to the previously outlined approach for defining consensus problems, Nakamoto neither presented a concise formal problem statement nor a concrete system model in which the problem is to be solved. Instead, a practical protocol design was outlined which was promptly followed by an actual prototype implementation in early 2009 [Bit]. To this date there exists no official formalized specification of the Bitcoin protocol.

We first place the focus on attack modeling and its relation to formalization efforts of Nakamoto consensus. Subsequently, we outline a brief excerpt of publications that present or expand upon more formal models of Bitcoin and Nakamoto consensus in more detail and present a (non-exhaustive) table of works that relate to the formalization of Nakamoto consensus.

A. Attack Modeling

Nakamoto's Argument: In the Bitcoin white paper Nakamoto provides an informal argument of the security properties of the protocol by modeling the probability for successful double spending attacks as two competing actors performing a *Binomial random walk* [Nak08a]. Nakamoto draws an analogy to the Gambler's Ruin problem [Coo09] and it is outlined that the probability of an attacker being able to catch up to the honest player decreases exponentially in the number of steps k, if the probability for honest processes taking the next step is greater than that of the attacker. While the topic of *consensus* itself is not actually addressed in the white paper, the following sentence suggest Nakamoto is aware of the Bitcoin protocol's relation to it:

> "Any needed rules and incentives can be enforced with this consensus mechanism." [Nak08a]

Nakamoto provides an informal claim that Bitcoin's fundamental mechanism provides a solution to the Byzantine generals problem in the cryptography mailing list [Nak08b], however the therein outlined protocol is probabilistic in regard to reaching *agreement* and flaws were later pointed out [GKL15].

Block Withholding: Nakamoto's initial model does not account for the possibility that an attacker may somehow *split* or *waste* the computational power of honest miners, thereby affecting the *validity* property of protocols based on Nakamoto consensus. Such an attack, referred to as *selfish mining*, is outlined by Eyal and Sirer [ES14]. The key idea behind this strategy is that an adversary initially keeps its discovered blocks private in an attempt to direct honest miners' computational power towards extending a chain, which will later be considered *stale*. The attack is successful in case the created private chain exceeds the public chain in terms of length and the adversary reveals its blocks, forcing honest miners into chain reorganization, i.e., to adopt the adversary's now public branch.

Eyal and Sirer model their system as a static set of miners $1, \ldots, n$, where each miner i has mining power m_i, such that $\sum_{i=1}^{n} m_i = 1$. Each miner chooses a chain head to mine, and finds a subsequent block for that head after a time interval that is exponentially distributed with mean m_i^{-1}. Thereby, honest miners are modeled to always extend the longest chain, while the attacker may deviate from protocol rules. The characteristics of the communication network are not explicitly outlined, however a parameter γ is used to denote the probability of honest miners accepting a block published by the adversary over that of an honest miner for the same height, dependent on network connectivity and in particular message propagation speed.

Based on this abstraction of the protocol, it is shown that Bitcoin is not incentive-compatible. Specifically, there exists a threshold of relative mining power after which an adversary engaging in selfish mining exceeds the expected relative revenue of honest mining. In this context, the upper bound on the threshold size is shown to be 1/3, i.e., an attacker with more than 1/3 of the total mining power will always stand to benefit from selfish mining.

Sapirshtein et at. later derive ϵ-optimal selfish mining strategies, lowering the requirements with regards to computational power and extend the initial selfish mining model to consider network delays [SSZ15]. In the latter scenario, it is argued any attacker can benefit from deviating from honest mining, although concrete gains and thresholds are not yet discussed.

Nayak et al. further introduce a new class of so called *stubborn* mining attacks, outperforming the original selfish mining strategy in terms of revenue gain in [NKMS16]. Furthermore, they extend the attack model by considering combinations of selfish mining and *eclipse* attacks as introduced by Heilman et al. in [HKZG15], i.e., adversarial partitioning of the network to control communication between segments. Gervais et al. in turn are able to show how an adversary can deny the delivery of blocks to a subset of honest participants without network partitioning in [GRKC15] .

The described attack models are further enhanced by Gervais et al. to account for mining costs in the adversary's utility function in [GKW$^+$16]. Furthermore, a detailed empirical and simulation analysis of block interval and size parametrization impacts on the success thresholds of selfish mining and double spending attacks is provided.

B. Formal Analyses and Definitions of Nakamoto Consensus

There is currently no agreed-upon definition of Nakamoto consensus. One may only consider individual aspects of the Bitcoin protocol, or its entirety. However, the term is finding its way into more and more publications and we believe it is a suitable descriptor for this novel consensus approach. A fundamental difficulty in providing a good generalized definition lies in the tight interaction between the various mechanisms that make up the Bitcoin protocol. We outline formalization approaches of the Bitcoin protocol in a time-ordered fashion (taking into account that pre-prints and e-prints were sometimes published much earlier than the peer-reviewed publications) to paint a coherent picture how the field has evolved over the recent years.

Single-Shot anonymous Byzantine consensus: In [MJ14] Miller and LaViola give the first formal presentation of Bitcoin's consensus mechanism in the context of *single-shot* instances of Byzantine consensus. Specifically, they relate it to both *Monte Carlo* randomized consensus, that is, *probabilistic* consensus where there is a negligible but non-zero probability of error on agreement and *anonymous* consensus. Their system model assumes a static set of processes $\{p_1, p_2, \dots, p_n\} = \Pi$ where communication and processing time is assumed to be synchronous and reliable; however, processes have no way of determining message origins. The adversary is non-adaptive and given control over f of the processes, which she must designate at the outset. Each process may query a *random oracle* at most once per round which serves as an abstraction for the proof-of-work computational puzzle, thereby assuming processes each have identical computing power q. The model also assumes an arbitrary set of passive clients in addition to the n consensus participants which can only receive messages and *decide* on an output v.

Monte Carlo Consensus: A (binary) Monte Carlo consensus protocol for a set of n processes (f of which may be corrupted) begins with each correct process p_i receiving an input value $proposed_i \in \{0,1\}^*$, and must satisfy the following properties:

1) **Termination:** All clients must decide in bounded time.
2) **Agreement:** All correct processes must decide on the same value, except with *negligible probability*.
3) **Unanimity:** The decided value must be one of the inputs (*with non-negligible probability*).

Instead of constructing a concatenated blockchain, processes exchange their preferences with proofs-of-work and adopt, as their own preference, the value that appears to have the most votes. Under these assumptions, Miller and LaViola [MJ14] shows that such a *Bitcoin Consensus Protocol* can satisfy the presented Monte Carlo consensus properties for an adversary that controls strictly less than half, i.e., $\sum_{b\in\mathcal{B}(t)} m(b) < 50\%$ of the computing power. If there are no faults, then the expected number of distinct messages is $O(k)$, however in the worst case, when the number of Byzantine faults is $f = \lfloor\frac{n-1}{2}\rfloor$, it is shown that the expected message complexity grows to $O(kn^2)$.

Interestingly, this result regarding failure resilience for single shot consensus of what we may consider a variant of Nakamoto consensus does not easily translate to consensus for multiple instances, which is required when considering a blockchain data structure or different system states for state machine replication. In a multiple instance model, adversaries may adopt certain strategies such as block-withholding attacks [ES14], [NKMS16] that are not relevant in the single instance consensus model.

The Backbone protocol: In [GKL15] Garay et al. present the first formalization of fundamental principles behind the Bitcoin protocol, which they refer to as the *Bitcoin backbone*.

Its characteristics are described through two properties, namely *common prefix* and *chain-quality*. The assumed system model is synchronous and static with reliable communication channels, where processes cannot authenticate each other, implying a weaker message model than *oral messages*. The analysis considers trust assumptions where an *adaptive* and *rushing* adversary controls at most $f < \frac{n}{2}$ processes, which corresponds to the honest majority of hash-rate initially assumed by Nakamoto.[7] Rushing means that the adversary knows the messages of all honest users in each round before he has to decide on his strategy. Adaptive means that the adversary can take control of different processes "on the fly" provided it are at most f processes at any point in time. All of the n participants are allowed the same number of queries q per round to a *random oracle* as an abstraction for the hash function used in hash based proof-of-work, and it is assumed that the adversary has $f \cdot q$, $f < n$, such queries. The backbone protocol is parametrized by three external functions, which can be used to specify applications building on top of it.

Garay et al. base their analyses of the presented protocols on a multiparty setting employing elements from Canetti [Can00], [Can01] and Katz et al. [KMTZ13].

Specific applications of the backbone protocol are given by providing analyses for *Byzantine agreement* and *public*

[7]For reasons of self consistency this paper uses f to refer to the number of faulty nodes, wheres Garay et al. use the variable t instead of f

transaction ledgers based on top the backbone protocol. A *robust transaction ledger* is characterized by the two properties *persistence* and *liveness*. Persistence states that once a transaction goes more than k blocks "deep" into the blockchain of one honest player, then it will be permanently included in the blockchain of every honest with overwhelming probability. Liveness states that all (valid) transactions will eventually end up at a depth more than k blocks in an honest player's blockchain, and hence the adversary cannot selective deny the inclusion of transactions with high probability.

Persistence and liveness can be derived from the two main properties of the underlying backbone protocol *common-prefix* and *chain quality*.

The common-prefix and chain-quality properties are quantified by three parameters, γ, β and m, where γ and β correspond to the collective hashing power per round of honest nodes and the adversary respectively, and where m represents the expected number of PoWs that may be found per round by the participants as a whole.[8] The parameter m plays a critical role and should be small, i.e close to $\frac{n}{2}$, so the network synchronizes significantly faster than the rate of finding PoWs. If if gets close to 1, i.e the network desynchronizes, achieving a provable common prefix requires almost all participants to be honest.

Definition 5: Properties of the backbone protocol[9]

1) **common-prefix:** If $\frac{f}{n-f}$ is suitably bounded below 1, then the local sequence of blocks of honest players has a large common prefix, i.e. the probability for two honest processes to maintain mutually equal prefixes of their blockchains by removing k blocks from the top of their local chains increases exponentially in k.
2) **chain quality:** It is shown that the ratio of an adversary's blocks in the chains of honest processes are bounded by $\frac{f}{n-f}$, nevertheless it can be strictly bigger than β. If an adversary controls less than $\frac{1}{3}$ hashing power, i.e. $f < \frac{n}{3}$ it is shown that it will provably control less than 50% of the blocks in the honest processes' chains. *Ideal chain-quality* defines that the ratio of an adversary's blocks is exactly proportional to its hash power $\frac{f}{n}$.

Garay et al. analyze an instantiation of Byzantine Agreement (BA), namely *randomized anonymous byzantine binary consensus*, based on the backbone protocol. Specifically, they outline how the resulting protocol derives the *agreement* and *validity* properties of consensus (with *high probability*) from the common-prefix and chain-quality properties of the underling backbone protocol, while the *termination* property of consensus is more informally argued to be satisfied.

Definition 6: BA based on the backbone protocol

1) **Agreement:** There is a round after which all honest parties return the same output if queried by the environment.
2) **Validity:** The output returned by an honest party P equals the input of some party P' that is honest at the round Ps output is produced.

Without modifications, the resulting backbone BA protocol requires the adversary to control $f < \frac{n}{3}$ of the hash power in order for the validity property to hold with overwhelming probability. Else, expanding upon initial observations from Miller and LaViola in [MJ14], if an adversary has close to $f < \frac{n}{2}$ hash power, *validity* can only be guaranteed with *constant probability*. A solution to this problem for the q-bounded setting is presented through the introduction of a so called *2-for-1 PoW*, which turns two distinct and independent oracles into a protocol that only requires a single oracle. Through this 2-for-1 PoW a backbone-based BA algorithm is outlined where validity can be ensured with overwhelming probability if $f < \frac{n}{2}$. Garay et al. also point out that **strong validity** for *multi-valued* BA, where the input is chosen from set V, can be ensured if the hash power of adversary is restricted to $\frac{1-\delta}{|V|}$.

For the instantiation of a *public transaction ledger* we informally outline the *persistence* and *liveness* properties that are shown to hold if $f < \frac{n}{2}$. *Persistence* ensures that once a transaction is included at least k blocks deep in the sequence of blocks then any honest process will report the transaction as stable and in the same position with overwhelming probability. *Liveness* guarantees that if a transaction is provided as input to all honest players continuously for some u consecutive rounds, then all honest parties will report this transaction at least k blocks from the end of the ledger. Garay et al. outline that the proofs for these properties does not show that all of Bitcoin's objectives are met and, in particular, participant's *incentives* are not taken into account.

We point out that while Byzantine agreement based on the backbone protocol requires additional modifications to retain desirable validity properties for $f < \frac{n}{2}$, this is not the case for the instantiation of a public transaction ledger, thereby outlining the possibility that the problem of providing a transaction ledger[10] may be weaker than BA.

Speed-Security Tradeoffs: In a subsequent work Kiayias and Panagiotakos [KP15] extends upon the backbone protocol from [GKL15] with performance aspects in mind, to also capture questions regarding the security bounds of the backbone protocol under faster block generation rates. In the process of modeling and proving that the properties of a *robust transaction ledger* hold in this setting, it is recognized that a third property besides *common-prefix* and *chain-quality* would be helpful to provide a measure for speed. Therefore, they introduced a new property called *chain-growth*, which was only considered implicitly before that in [GKL15]. Note that an updated version of the e-print of [GKL15] also contains this third property.

Definition 7: **chain-growth:** This property requires that the chain of any honest participant grows by a factor of τ over time i.e., it holds that for any $s \in \mathbb{N}$ rounds, there are at least $\tau \cdot s$ blocks added to the chain. This allows to quantify the number of blocks that are added to the blockchain during any given number of rounds.

As a result, it is shown that the security guarantees of such protocols also hold for faster block generation rates

[8]Garay et al. use the variable name f instead of m.

[9]Note that, an updated version of the eprint also contains a third property called *chain growth* that was first introduced in [KP15].

[10]Of course depending on the problem definition and system model.

and degrades as the block generation rate increases to more than one block per round. A full communication round is assumed to take 12.6 seconds, which corresponds to the average Bitcoin block propagation time that was empirically derived in [DW13]. The overall system model (synchronous communication, static difficulty and hashrate) is kept the same as in the previously outlined *backbone model* [GKL15].

Bitcoin-NG: In [EGSvR16] Eyal et al. define Nakamoto consensus in the context of state machine replication. In their model the system is comprised of a set of processes $\{p_1, p_2, \ldots, p_n\} = \Pi$ connected by a reliable peer-to-peer network. Each process has access to a random bit source through a (cryptographic) *random oracle*. Processes can generate key pairs but no trusted PKI is assumed. A cryptographic proof-of-work scheme is assumed, where each process $p \in \Pi$ has limited computational power. The mining power of process p_i, denoted by $m(i)$, is the number of attempts per second a given process can make when searching for a solution to the PoW with respect to its limited compute power. At any time t a subset of nodes $\mathcal{B}(t) \subset \Pi$ are Byzantine where, based on the previous findings on selfish mining by Eyal and Sirer [ES14], they assume an upper bound on the combined mining power of $\mathcal{B}(t)$ at any time t that is:

$$\forall t : \sum_{b \in \mathcal{B}(t)} m(b) < \frac{1}{4} \sum_{p \in \Pi} m(p)$$

Or, in other words, the combined mining power of Byzantine nodes at any time is required to be less than $\frac{1}{4}$. In their model Nakamoto consensus is expressed through the following three properties:

Definition 8: Properties of Nakamoto consensus as by Eyal et al. [EGSvR16]

1) **Termination:** There exists a time difference function $\Delta(\cdot)$ such that, given a time t and a value $0 < \varepsilon < 1$, the probability is smaller than ε that at times $t', t'' > t + \Delta(\varepsilon)$ a node returns two different states for the machine at time t.
2) **Agreement:** There exists a time difference function $\Delta(\cdot)$ such that, given a $0 < \varepsilon < 1$, the probability that at time t two nodes return different states for $t - \Delta(\varepsilon)$ is smaller than ε.
3) **Validity:** If the fraction of mining power of Byzantine nodes is bounded by f, i.e., $\forall t : \frac{\sum_{b \in \mathcal{B}(t)} m(b)}{\sum_{p \in \Pi} m(p)} < f$, then the average fraction of state machine transitions that are not inputs of honest nodes is smaller than f.

Analysis in Δ-bounded Networks: In [PSs16] Pass et al. analyze and formally prove that the blockchain consensus mechanism satisfies strong forms of *consistency* and *liveness* in networks with *a-priori* Δ-bounded delays, i.e. *partially synchronous* networks, where the adversary may delay the delivery of any message for as long as Δ. Specifically, as long as the adversary controls less than half of the computational power ($\rho < \frac{1}{2}$), for every Δ there exists some sufficiently small mining-hardness, $p \leq \frac{1}{\rho n \Delta}$, such that blockchain consensus satisfies *consistency*. They also show that in fully asynchronous network models consistency can not be satisfied without an upper bound Δ on the network delay, even if the adversary only controls a tiny fraction of computational power.[11]

The assumed system model uses a random oracle as an abstraction for the computational puzzles analogous to [GKL15] and considers *adaptive corruption* of participants. The network is assumed to be fully connected, reliable, and messages are delivered within Δ rounds.

It is furthermore remarked that since protocol security is shown to exist for some mining-hardness parameter p for every n, Δ, only a *very rough upper-bound* on the number of participants n needs to be known, thereby suggesting the ability to allow for a *dynamic* system model. However, a more formal analysis of the dynamic setting is left for future work.

Additionally, an abstract notion of a blockchain protocol that should satisfy the following four key properties is formally defined:

Definition 9: Abstract blockchain

1) **consistency:** with overwhelming probability (in T)[12], at any point, the chains of two honest players can differ only in the last T blocks.
2) **future self-consistence:** with overwhelming probability (in T), at any two points r, s the chains of any honest player at r and s differ only within the last T blocks.
3) **g-chain-growth:** with overwhelming probability (in T), at any point in the execution, the chain of honest players grows by at least T messages in the last $\frac{T}{g}$ rounds; g is called the chain-growth of the protocol.
4) the **μ-chain quality:** with overwhelming probability (in T), for any T consecutive messages in any chain held by some honest player, the fraction of messages that were "contributed by honest players" is at least μ.

It is demonstrated that any blockchain protocol satisfying these properties can be used as the basis for a *public ledger* that satisfies *persistence* and *liveness* properties, such as those outlined by Garay et al. [GKL15], in a black-box manner.

On Trees, Chains and Fast Transactions: In [KP16] Kiayias and Panagiotakos extends the upon the backbone model from [GKL15], [KP15] to also include blockchain protocols with more tree-like structures such as GHOST [SZ13]. Thereby, they show that GHOST (trees) as well as Bitcoin (chains) can be described in a new analysis framework that reduces the properties of the *robust transaction ledger*, i.e. common-prefix, chain quality, to a single lemma called the *fresh block lemma*. Thereby, this paper presents an orthogonal proof strategy compared to [GKL15], [KP15], [PSs16].

Definition 10: **Fresh Block Lemma:** At any point of the execution and for any past sequence of s consecutive rounds, there exists an honest block mined in these rounds, that is contained in the chain of any honest player from this point on.

Using this lemma, the backbone model from [GKL15] is adapted to account for trees and is called the *GHOST backbone*, and it is proven that both GHOST and Bitcoin implement

[11]Specifically when $\Delta = \frac{1+\delta}{\rho n p}$ for some $\delta > 0$, where $\rho n p$ is the expected number of blocks that an attacker can mine in a round.

[12]T is comparable to the parameter k of the backbone protocol and represents the number of "unconfirmed" blocks for which agreement does not have to hold with high probability

a robust transaction ledger in the q-bounded synchronous setting that satisfies persistence and liveness.

Sleepy Consensus in [BPS16a] Pass et al. introduce the concept of a *"sleepy"* model of computation for players that extends upon the more classical differentiation of *honest* and *corrupted / adversarial* participants, where players can either be *online* (alert) or *offline* (asleep). It is argued that for "large-scale" consensus protocols with potentially millions of participants, such as blockchain protocols, an always online assumption is unrealistic. Their approach relates to previous work by Micali on the construction of a distributed ledger, where honest majority assumptions are also only extended towards *currently active* participants [Mic16].

The key question that Pass et al. positively show to be achievable through formal analysis is: *"Can we design a consensus protocol that achieves consistency and liveness assuming only that a* majority of the online players *are honest?"*

In this respect the presented formalization and proofs leverage on the previous analysis of the Bitcoin protocol in Pass et al. [PSs16], however the reliance on proofs-of-work is dispensed of and a static set of participants, i.e. a *permissioned* model, assumed. Specifically, the proofs-of-work are replaced by redefining the puzzle solution to be of the form $(\mathcal{P}, t)$, where $\mathcal{P}$ is the player's identifier, t is the block time, and a valid puzzle solution is $H(\mathcal{P}, t) < \mathcal{D}_p$, where H is a random oracle and $\mathcal{D}_p$ is a parameter such that the hash outcome is only smaller than $\mathcal{D}_p$ with probability p. To protect against adversaries the following two additional restrictions are imposed:

1) A valid chain must have strictly increasing block-times
2) A valid chain cannot contain any block-times for the "future" (where "future" is adjusted to account for nodes' clock offsets)

Furthermore, to overcome the problem of adaptive corruption and sleepiness when leader election is predictable, inspiration is taken from Micali [Mic16]. Players each commit to a secret seed for the random oracle H[13] which enables players to privately evaluate the puzzle solution and also prove to other participants in zero-knowledge that the oracle was correctly evaluated. If leaders only publish this proof as part of their message an adversary has no better way of trying to corrupt the next leader than to randomly select from the set of participants.

Their system model relies, among other, on a Public-Key-Infrastructure (PKI), the existence of a Common Random String (CRS) and collision-resistant hash functions. A partially synchronous timing model analogous to Interactive Turing Machine (ITM) model [Can01] is considered and the communication network is fully connected and reliable.

Two Protocols are presented and shown to satisfy the following theorems under the assumption that a majority of the *awake* players are honest:

- **Theorem 1.** Assume the existence of families of a collision-resistant hash functions (CRH). Then, there exists a protocol for state-machine replication in the Bare PKI, CRS and in the timing model, which achieves consistency and liveness assuming a static online schedule and static corruptions, as long as at any point in the execution, a majority of the awake players are honest.
- **Theorem 2.** Assume the existence of families of sub-exponentially secure collision-resistant hash functions (CRH), and enhanced trapdoor permutations (TDP). Then, there exists a state-machine replication protocol in the Bare PKI, CRS and timing model, which achieves consistency and liveness under adaptive corruptions as long as at any point in the execution, a majority of the awake players are honest.

[13]Which can be instantiated through a Pseudo-random function (PRF).

Chains of variable difficulty: In [GKL16] Garay et al. adapts the backbone model from [GKL15] to the *dynamic q-bounded synchronous setting* to account for difficulty adjustments. Thereby, they show that a Bitcoin-like target recalculation function satisfies common prefix and chain quality and consequently implements a *robust transaction ledger* if the change in the number of parties is bounded over a certain number of rounds, i.e, (γ, s)*-respecting*. The chain-growth property is modeled in a lemma and referes to difficulty instead of number of blocks compared to [KP15]. Informally, it states that honest parties will make progress proportional to the PoW they obtain, which ensures that honest parties advance in terms of accumulated difficulty despite all possible actions of an adversary. For their proof, they first define a *typical* execution in which the successes of the adversary and honest parties do not deviate too much from their expectations. Then they show that almost all polynomially bounded execution are typical. As a result they note that the length of a difficulty adjustment epoch ($m = 2016$ in Bitcoin) is a security parameter that should be large to provide a sufficiently small probability of attack.

In this paper, the underlying hash function $H(\cdot)$ is modeled as a random oracle to exclude the unlikely cases that a *bad event*, i.e., a collision, happens in their model. This provides the motivation for the follow up paper [GKP17].

V. New and Hybrid Consensus Models inspired by Nakamoto Consensus

This section gives a brief outlook of some new and hybrid approaches that try to unite desirable properties from classical BFT- as well as Nakamoto consensus. On the one hand, these are Proof-of-Stake based approaches such as *Ouroboros*,*Algorand*, and *Snow White*, which are aimed at avoiding the resource intensive Proof-of-Work mechanism, on the other hand, we see proposals such as *Thunderella*, which intends to gain the resilience and permissionlessness of Nakamoto consensus while leveraging on the fast confirmation/consensus finality of traditional BFT.

In [SZ13] Sompolinsky and Zohar introduce GHOST (Greedy Heaviest-Observed Sub-Tree) as a new branch selection policy, which evaluates each chain's weight rather than length and allows to account for stale blocks, aiming at reducing the time to converge to a consistent global state. In this context, they model the network as a directed graph $G = (V, E)$, where the edges' values represent the network propagation delay between adjacent nodes in V. In [LSZ15]

Lewenberg et al. extends upon this concept and proposes to restructure the blockchain into a directed acyclic graph (DAG), increasing tolerance for large blocks with longer propagation times and in turn possibly achieving a higher transaction volume. Consequently, in [SLZ16] the same authors present SPECTRE, a new DAG-based consensus protocol facilitating significantly faster transaction confirmation times than in Bitcoin. However, in contrast to Nakamoto consensus, where conflicts can be resolved between any pair of transactions by deciding upon their ordering, SPECTRE can only provide probabilistic guarantees to some limited set of transaction pairs. As such, the time to identify the prevailing transaction in a conflict is said to potentially be arbitrary long.

Micali et al. presents Algorand, a novel approach for a (permissionless) Proof-of-Stake distributed ledger[Mic16], [GHM$^+$17]. By leveraging on the *player replaceability* property of a novel BA protocol (*BA*$\star$) [Mic17] and a leader/verifier selection algorithm, which is executed privately, Algorand is able to withstand a powerful adaptive adversary. This construction allows for a strong notion of consistency, where the probability for disagreement on the underlying ledger, rather than being dependent on a security parameter in the number of blocks "confirming" a particular state such as [GKL15], [PSs16], is a configurable protocol parameter.[14] The efficiency of the protocol is based both on *BA*$\star$, which the authors describe as "the most efficient cryptographic BA protocol for SC [synchronous complete] networks known so far", and the fact that the protocol is only executed by a randomly selected subset of participants.

In [KRDO16] Kiayias et al. introduces Ouroboros, one of the first Proof-of-Stake protocols based on a rigorous formal security analysis that extends upon the previously outlined backbone model of Garay et al. The paper first outlines a *static stake* model, where leaders are assigned to blockchain slots with probability proportional to their fixed stake, which is later extended to the *dynamic stake* model. In both cases the authors first abstract the leader selection process by an *ideal function* that is later instantiated with a specific PVSS based scheme. With Ouroboros Praos [DGKR17], the model is extended to partially synchronous network settings, while also accounting for an adaptive adversary.

Concurrently, Bentov et al. [BPS16b] presents *Snow White*, a Proof-of-Stake blockchain protocol that extends upon the previously outlined *sleepy model* of Pass et al. [BPS16a]. In particular, the *permissionless* setting is considered by introducing a mechanism that relies on the distribution of some form of stake, i.e. cryptographic currency units, for the periodical rotation of the consensus committee.

In traditional Nakamoto consensus based blockchains, a minority attacker can gain rewards higher than their fair share by utilizing selfish mining strategies. The Fruitchain protocol [PS16a], presented by Pass and Shi, introduces a new way of distributing rewards, such that miners are guaranteed a share of the overall reward approximate to their computational power, while maintaining identical consistency and liveliness properties as Bitcoin's Nakamoto consensus. Thereby, transactions are stored in so called *fruits*, which are linked to blocks and require miners to perform two separate Proofs-of-Work in parallel[15]. Because the ability to include old, i.e., stale fruits has to be time-bounded, fairness is only guaranteed to within some parameterizable ϵ.

Thunderella [PS17] is an improvement proposal for permissionless (PoW) blockchain protocols that leverages on the notion of *optimistic responsiveness* to provide both, a fast asynchronous path, as well as a synchronous fall-back mechanism in case certain assumptions no longer hold. The basic idea is to use the underlying blockchain's Nakamoto consensus to bootstrap a special player, referred to as the *accelerator*, who facilitates faster transaction confirmations. Under the optimistic assumption that $\frac{3}{4}$ of the computational power as well as the accelerator is honest, transactions are confirmed as fast as the actual message delay in the network, providing a fast path. Otherwise, the protocol falls back to the regular slow path (with honest majority hashrate assumption) until, eventually, conditions for the optimistic fast path can be re-established.

VI. Concluding Remarks

It is becoming evermore clear that Bitcoin, and distributed ledger technologies in general, are having a significant impact on various research communities. In this regard we would like to draw an analogy to the early research on the consensus problem that was outlined at the beginning of this work. Based on the desire to solve a practical problem, namely the synchronization of clocks, stabilization of input from sensors, and agreement on results of diagnostic tests, Pease et al. [PSL80] identified a fundamental problem in fault-tolerant and distributed computing. Similarly, Bitcoin presents a practical solution to an extension of this fundamental consensus problem, namely how to reach agreement in a large-scale *decentralized* system if some participants behave arbitrarily or even maliciously.

Understanding the fundamental mechanisms and security properties of this new class of Nakamoto consensus algorithms is proving to become increasingly important, as their utilization is being considered for a broad range of applications.

In consideration of the many new proposals for alternative consensus mechanisms and distributed ledger designs that have not yet experienced as much rigorous security analysis as Bitcoin has endured, our analysis of the current research efforts towards the formalization of Nakamoto consensus will hopefully aid in bridging this gap, and motivate protocol designers to work closely together with the computer security and distributed systems research community.

[14][Mic16] considers a parametrization corresponding to a failure probability of $F = 10^{-12}$ and $F = 10^{-18}$.

[15]This is achieved by the 2-for-1 trick described by Garay et al. [GKL16]

TABLE I. NON-EXHAUSTIVE TIMELINE OF PAPERS RELATED TO THE ANALYSIS OF NAKAMOTO CONSENSUS

Date[†]	Title	Ref.	Category	Overview
2008 Dec	Bitcoin: A Peer-to-Peer Electronic Cash System	[Nak08a]	consensus protocol, PoW	Initial informal description of Nakamoto style consensus. Simulation via Binomial Random Walk.
2013 Nov	Majority is not enough: Bitcoin mining is vulnerable	[ES14]	security analysis	Introduces selfish mining attacks and shows Nakamoto consensus is not incentive compatible.
2013 Dec	Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains	[SZ13]	consensus protocol, PoW	Introduces GHOST, a new blockchain branch selection policy, which reduces the time to converge to a single consistent chain.
2014 Feb	Analysis of Hashrate-Based Double Spending	[Ros14]	security analysis	Evaluates protection against double-spending and derives success probabilities. Updates Nakamoto's original protocol description.
2014 Apr	Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin	[MJ14]	formalization	Gives the first formal presentation of Bitcoin's consensus mechanism in the context of *single-shot* instances of Byzantine consensus.
2014 Sep	The Bitcoin Backbone Protocol: Analysis and Applications	[GKL15]	formalization	Presents the first formalization of fundamental principles behind the Bitcoin protocol.
2015 Jun	Optimal selfish mining strategies in Bitcoin	[SSZ15]	security analysis	Derives ϵ-optimal selfish mining strategies, lowering the attack threshold with regards to computational power and extends the initial selfish mining model to consider network delays.
2015 Jan	Inclusive Blockchain Protocols	[LSZ15]	consensus protocol, PoW, DAG	Proposes a DAG-based chain structure, allowing blocks to reference multiple predecessors this way potentially increasing the transaction volume.
2015 Mar	Eclipse Attacks on Bitcoin's Peer-to-Peer Network	[HKZG15]	security analysis	Describes eclipse attacks on Bitcoin's peer-to-peer network, where an adversary partitions the network to control communication between segments.
2015 Jun	Tampering with the Delivery of Blocks and Transactions in Bitcoin	[GRKC15]	security analysis	Shows issues, potential attacks and countermeasures of Bitcoin's current optimizations and scalability measures.
2015 Jul	Optimal selfish mining strategies in Bitcoin	[SSZ15]	security analysis	Derives ϵ-optimal selfish mining policies and evaluates selfish mining in the presence of communication delays.
2015 Aug	Stubborn mining: Generalizing selfish mining and combining with an eclipse attack	[NKMS16]	security analysis	Introduces stubborn mining strategies, outperforming selfish mining, and composed mining/network level eclipse attacks.
2015 Oct	Speed-Security Tradeoffs in Blockchain Protocols	[KP15]	security analysis, formalization	Extends upon the backbone protocol model and introduces the *chain growth* property.
2015 Oct	The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication	[Vuk15]	systematization	Outlines differences between "classical" BFT and Nakamoto consensus approaches.
2015 Oct	Bitcoin-NG: A Scalable Blockchain Protocol	[EGSvR16]	consensus protocol, PoW	Introduces Bitcoin-NG, where transactions are contained in microblocks, published by a leader/miner between consecutive normal or *key* blocks.
2016 Feb	The honey badger of BFT protocols	[MXC$^+$16]	consensus protocol, BFT	Describes a new (randomized) BFT consensus protocol specifically targeted towards an application for blockchains/many participants.
2016 May	Analysis of the Blockchain Protocol in Asynchronous Networks	[PSs16]	formalization	Shows blockchain consensus mechanisms satisfy strong forms of consistency and liveness in partially sync. networks. Introduces the abstract notion of a blockchain with *consistency*, *future self-consistence*, *g-chain-growth* and *μ-chain quality* properties.
2016 May	Bitcoin's Security Model Revisited	[SZ16]	security analysis	Analyzes Bitcoin's security in regard to double spending attacks.
2016 Jun	On Trees, Chains and Fast Transactions in the Blockchain	[KP16]	formalization	Presents an extension of the backbone model to include tree-like protocols, such as GHOST.
2016 Jun	On the Security and Performance of Proof of Work Blockchains	[GKW$^+$16]	security analysis	Extends selfish mining and double spending models to account for mining costs. Provides empirical and simulation analysis of the impacts of block interval and size parametrization on attack success thresholds.
2016 Jul	ALGORAND: The Efficient and Democratic Ledger	[Mic16]	consensus protocol, PoS, BFT	Describes a PoS blockchain protocol using novel BA protocol with private leader/verifier selection in an adaptive adversary model.
2016 Sep	Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol	[KRDO16]	consensus protocol, PoS	Introduces a new PoS blockchain protocol and provides formal security proofs that extend upon the backbone model.
2016 Sep	The sleepy model of consensus	[BPS16a]	formalization	Introduction of the *sleepy model* of consensus and a consensus protocol which achieves consistency and liveness, given the majority of online nodes are behaving honestly. Builds upon [PSs16].
2016 Sep	Hybrid Consensus: Scalable Permissionless Consensus	[PS16b]	consensus protocol, PoW, BFT	Analyzes blockchain performance limits, introduces of the *responsiveness* property and describes a responsive consensus protocol in the permissionless setting.
2016 Sep	Snow White: Provably Secure Proofs of Stake	[BPS16b]	consensus protocol, PoS	Analyzes requirements for PoS blockchain protocols and presents a new approach which provably archvies these requirements, extending upon [BPS16a].
2016 Sep	Fruitchains: A Fair Blockchain	[PS16a]	consensus protocol, PoW	Introduces a new way of distributing rewards, such that miners are guaranteed a share of the overall reward approximate to their computational power, thereby addressing the issue of selfish mining.
2016 Nov	The Bitcoin Backbone Protocol with Chains of Variable Difficulty	[GKL16]	formalization	Adopts the backbone protocol to the *dynamic q-bounded synchronous setting* to account for difficulty adjustments
2016 Dec	SPECTRE: A Fast and Scalable Cryptocurrency Protocol	[SLZ16]	consensus protocol, PoW, DAG	Introduces a new DAG-based consensus protocol, focusing on faster transaction confirmation times.
2016 Dec	Solidus: A Blockchain Protocol Based on Reconfigurable Byzantine Consensus	[AMN$^+$16]	consensus protocol, PoW, BFT	Presents a new protocol, which uses a combination of PoW and PBFT to achieve scalable permissionless Byzantine consensus
2017 Jun	Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol	[DGKR17]	consensus protocol, PoS	Extends [KRDO16], considering semi-synchronous communication and an adaptive adversary model.
2017 Jul	Blockchain Consensus Protocols in the Wild	[CV17]	security analysis	Outlines the necessity to formalize trust assumptions and security guarantees as well as analyzes deployed blockchain protocols.
2017 Sep	Thunderella: Blockchains with Optimistic Instant Confirmation	[PS17]	consensus protocol, PoW	Introduces a PoW blockchain protocol with optimistic fast confirmations in case of 3/4 honest nodes and honest leader.

[†]Dates listed correspond to the time the publication was first made available, i.e., in some cases as a pre-print, to paint a more coherent picture.

References

[AAC+05] Amitanand S Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *ACM SIGOPS operating systems review*, volume 39, pages 45–58. ACM, 2005.

[ABSFG08] Eduardo A Alchieri, Alysson Neves Bessani, Joni Silva Fraga, and Fabíola Greve. Byzantine consensus with unknown participants. In *Proceedings of the 12th International Conference on Principles of Distributed Systems*, pages 22–40. Springer-Verlag, 2008.

[AFJ06] Dana Angluin, Michael J Fischer, and Hong Jiang. Stabilizing consensus in mobile networks. In *Distributed Computing in Sensor Systems*, pages 37–50. Springer, 2006.

[AJK05] James Aspnes, Collin Jackson, and Arvind Krishnamurthy. Exposing computationally-challenged byzantine impostors. *Department of Computer Science, Yale University, New Haven, CT, Tech. Rep*, 2005.

[AMN+16] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus. https://arxiv.org/abs/1612.02916, Dec 2016. Accessed: 2017-02-06.

[AS98] Yair Amir and Jonathan Stanton. The spread wide area group communication system. Technical report, TR CNDS-98-4, The Center for Networking and Distributed Systems, The Johns Hopkins University, 1998.

[Bag00] Walter Bagehot. *The english constitution*, volume 3. Kegan Paul, Trench, Trübner, 1900.

[Ban98] Bela Ban. Design and implementation of a reliable group communication toolkit for java, 1998.

[BBRTP07] Roberto Baldoni, Marin Bertier, Michel Raynal, and Sara Tucci-Piergiovanni. Looking for a definition of dynamic distributed systems. In *International Conference on Parallel Computing Technologies*, pages 1–14. Springer, 2007.

[Bit] Bitcoin community. Bitcoin-core source code. https://github.com/bitcoin/bitcoin. Accessed: 2015-06-30.

[BJ87] Ken Birman and Thomas Joseph. Exploiting virtual synchrony in distributed systems. volume 21. ACM, 1987.

[BMC+15] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2015.

[BO83] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30. ACM, 1983.

[BPS16a] Iddo Bentov, Rafael Pass, and Elaine Shi. The sleepy model of consensus. https://eprint.iacr.org/2016/918.pdf, 2016. Accessed: 2016-11-08.

[BPS16b] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. https://eprint.iacr.org/2016/919.pdf, 2016. Accessed: 2016-11-08.

[BR09] François Bonnet and Michel Raynal. The price of anonymity: Optimal consensus despite asynchrony, crash and anonymity. In *Proceedings of the 23rd international conference on Distributed computing*, pages 341–355. Springer-Verlag, 2009.

[Bre00] EA Brewer. Towards robust distributed systems. abstract. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, page 7, 2000.

[BSAB+17] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. arXiv:1711.03936, 2017. Accessed:2017-12-11.

[BT16] Zohir Bouzid and Corentin Travers. Anonymity-preserving failure detectors. In *International Symposium on Distributed Computing*, pages 173–186. Springer, 2016.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[CGR07] Tushar D Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 398–407. ACM, 2007.

[CGR11] Christian Cachin, Rachid Guerraoui, and Luis Rodrigues. *Introduction to reliable and secure distributed programming*. Springer Science & Business Media, 2011.

[CKS00] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, pages 123–132. ACM, 2000.

[CL+99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[CL02] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[CNV04] Miguel Correia, Nuno Ferreira Neves, and Paulo Verissimo. How to tolerate half less one byzantine nodes in practical distributed systems. In *Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on*, pages 174–183. IEEE, 2004.

[Coo09] J. L. Coolidge. The gambler's ruin. *Annals of Mathematics*, 10(4):181–192, 1909.

[Cri91] Flaviu Cristian. Reaching agreement on processor-group membrship in synchronous distributed systems. *Distributed Computing*, 4(4):175–187, 1991.

[CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. volume 43, pages 225–267. ACM, 1996.

[CV17] Christian Cachin and Marko Vukolić. Blockchain consensus protocols in the wild. arXiv:1707.01873, 2017. Accessed:2017-09-26.

[CVL10] Miguel Correia, Giuliana S Veronese, and Lau Cheuk Lung. Asynchronous byzantine consensus with 2f+ 1 processes. In *Proceedings of the 2010 ACM symposium on applied computing*, pages 475–480. ACM, 2010.

[CVNV11] Miguel Correia, Giuliana Santos Veronese, Nuno Ferreira Neves, and Paulo Verissimo. Byzantine consensus in asynchronous message-passing systems: a survey. volume 2, pages 141–161. Inderscience Publishers, 2011.

[CWA+09] Allen Clement, Edmund L Wong, Lorenzo Alvisi, Michael Dahlin, and Mirco Marchetti. Making byzantine fault tolerant systems tolerate byzantine faults. In *NSDI*, volume 9, pages 153–168, 2009.

[DDS87] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. volume 34, pages 77–97. ACM, 1987.

[Dei] Wei Dei. b-money. http://www.weidai.com/bmoney.txt. Accessed on 03/03/2017.

[DGFGK10] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Anne-Marie Kermarrec. Brief announcement: Byzantine agreement with homonyms. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pages 74–75. ACM, 2010.

[DGG02] Assia Doudou, Benoît Garbinato, and Rachid Guerraoui. Encapsulating failure detection: From crash to byzantine failures. In *International Conference on Reliable Software Technologies*, pages 24–50. Springer, 2002.

[DGKR17] Bernardo David, Peter Gaži, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. Cryptology ePrint Archive, Report 2017/573, 2017. Accessed: 2017-06-29.

[DLP$^+$86] Danny Dolev, Nancy A Lynch, Shlomit S Pinter, Eugene W Stark, and William E Weihl. Reaching approximate agreement in the presence of faults. volume 33, pages 499–516. ACM, 1986.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. volume 35, pages 288–323. ACM, 1988.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[Dol81] Danny Dolev. Unanimity in an unknown and unreliable environment. In *Foundations of Computer Science, 1981. SFCS'81. 22nd Annual Symposium on*, pages 159–168. IEEE, 1981.

[Dou02] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[DSU04] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, 36(4):372–421, 2004.

[DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[EGSvR16] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Security Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, Mar 2016.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[Fin04] Hal Finney. Reusable proofs of work (rpow). http://web.archive.org/web/20071222072154/http://rpow.net/, 2004. Accessed: 2016-04-31.

[Fis83] Michael J Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *International Conference on Fundamentals of Computation Theory*, pages 127–140. Springer, 1983.

[FL82] Michael J FISCHER and Nancy A LYNCH. A lower bound for the time to assure interactive consistency. volume 14, Jun 1982.

[FLP85] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. volume 32, pages 374–382. ACM, 1985.

[Fuz08] Rachele Fuzzati. *A formal approach to fault tolerant distributed consensus*. PhD thesis, EPFL, 2008.

[GHM$^+$17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. Accessed: 2017-06-29.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[GKL16] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. http://eprint.iacr.org/2016/1048.pdf, 2016. Accessed: 2017-02-06.

[GKP17] Juan A. Garay, Aggelos Kiayias, and Giorgos Panagiotakos. Proofs of work for blockchain protocols. Cryptology ePrint Archive, Report 2017/775, 2017. http://eprint.iacr.org/2017/775.

[GKQV10] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. In *Proceedings of the 5th European conference on Computer systems*, pages 363–376. ACM, 2010.

[GKTZ12] Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. pages 561–572. Springer, 2012.

[GKW$^+$16] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. https://eprint.iacr.org/2016/555.pdf, 2016. Accessed: 2016-08-10.

[GL02] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. volume 33, pages 51–59. ACM, 2002.

[GRKC15] Arthur Gervais, Hubert Ritzdorf, Ghassan O Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 692–705. ACM, 2015.

[Her88] Maurice P Herlihy. Impossibility and universality results for wait-free synchronization. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pages 276–290. ACM, 1988.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, 1991.

[HKZG15] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, 2015.

[Hoe07] Jaap-Henk Hoepman. Distributed double spending prevention. In *Security Protocols Workshop*, pages 152–165. Springer, 2007.

[HT94] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems. *Cornell University Technical Report 94-1425*, 1994.

[IT08] Hideaki Ishii and Roberto Tempo. Las vegas randomized algorithms in distributed consensus problems. In *2008 American Control Conference*, pages 2579–2584. IEEE, 2008.

[JB99] Ari Juels and John G Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS*, volume 99, pages 151–165, 1999.

[KMMS01] Kim Potter Kihlstrom, Louise E Moser, and P Michael Melliar-Smith. The securering group communication system. *ACM Transactions on Information and System Security (TISSEC)*, 4(4):371–406, 2001.

[KMMS03] Kim Potter Kihlstrom, Louise E Moser, and P Michael Melliar-Smith. Byzantine fault detectors for solving consensus. volume 46, pages 16–35. Br Computer Soc, 2003.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In *TCC*, volume 7785, pages 477–498. Springer, 2013.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoff s in blockchain protocols. https://eprint.iacr.org/2015/1019.pdf, Oct 2015. Accessed: 2016-10-17.

[KP16] Aggelos Kiayias and Giorgos Panagiotakos. On trees, chains and fast transactions in the blockchain. http://eprint.iacr.org/2016/545.pdf, 2016. Accessed: 2017-02-06.

[KRDO16] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. https://pdfs.semanticscholar.org/1c14/549f7ba7d6a000d79a7d12255eb11113e6fa.pdf, 2016. Accessed: 2017-02-20.

[Lam84] Leslie Lamport. Using time instead of timeout for fault-tolerant distributed systems. volume 6, pages 254–280. ACM, 1984.

[Lam98] Leslie Lamport. The part-time parliament. volume 16, pages 133–169. ACM, 1998.

[LCW$^+$06] Harry C Li, Allen Clement, Edmund L Wong, Jeff Napper, Indrajit Roy, Lorenzo Alvisi, and Michael Dahlin. Bar gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 191–204. USENIX Association, 2006.

[LSM06] Brian Neil Levine, Clay Shields, and N Boris Margolin. A survey of solutions to the sybil attack. *University of Massachusetts Amherst, Amherst, MA*, 7, 2006.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. volume 4, pages 382–401. ACM, 1982.

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

[LTKS15] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 706–719. ACM, 2015.

[Lyn96] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[Mic16] Silvio Micali. Algorand: The efficient and democratic ledger. http://arxiv.org/abs/1607.01341, 2016. Accessed: 2017-02-09.

[Mic17] Silvio Micali. Byzantine agreement, made trivial. https://people.csail.mit.edu/silvio/SelectedApr 2017. Accessed:2018-02-21.

[MJ14] A Miller and LaViola JJ. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. https://socrates1024.s3.amazonaws.com/consensus.pdf, 2014. Accessed: 2016-03-09.

[MMRT03] Dahlia Malkhi, Michael Merritt, Michael K Reiter, and Gadi Taubenfeld. Objects shared by byzantine processes. volume 16, pages 37–48. Springer, 2003.

[MPR01] Hugo Miranda, Alexandre Pinto, and Luıs Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Distributed Computing Systems, 2001. 21st International Conference on.*, pages 707–710. IEEE, 2001.

[MR97] Dahlia Malkhi and Michael Reiter. Unreliable intrusion detection in distributed computations. In *Computer Security Foundations Workshop, 1997. Proceedings., 10th*, pages 116–124. IEEE, 1997.

[MRT00] Achour Mostefaoui, Michel Raynal, and Frédéric Tronel. From binary consensus to multivalued consensus in asynchronous message-passing systems. *Information Processing Letters*, 73(5-6):207–212, 2000.

[MXC$^+$16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. https://eprint.iacr.org/2016/199.pdf, 2016. Accessed: 2017-01-10.

[Nak08a] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, Dec 2008. Accessed: 2015-07-01.

[Nak08b] Satoshi Nakamoto. Bitcoin p2p e-cash paper, 2008.

[Nar16] Narayanan, Arvind and Bonneau, Joseph and Felten, Edward and Miller, Andrew and Goldfeder, Steven. Bitcoin and cryptocurrency technologies. https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf?a=1, 2016. Accessed: 2016-03-29.

[Nei94] Gil Neiger. Distributed consensus revisited. *Information processing letters*, 49(4):195–201, 1994.

[NG16] Christopher Natoli and Vincent Gramoli. The blockchain anomaly. In *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*, pages 310–317. IEEE, 2016.

[NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *1st IEEE European Symposium on Security and Privacy, 2016*. IEEE, 2016.

[PS16a] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. http://eprint.iacr.org/2016/916.pdf, 2016. Accessed: 2016-11-08.

[PS16b] Rafael Pass and Elaine Shi. Hybrid consensus: Scalable permissionless consensus. https://eprint.iacr.org/2016/917.pdf, Sep 2016. Accessed: 2016-10-17.

[PS17] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. Cryptology ePrint Archive, Report 2017/913, 2017. Accessed:2017-09-26.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. volume 27, pages 228–234. ACM, 1980.

[PSs16] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. http://eprint.iacr.org/2016/454.pdf, 2016. Accessed: 2016-08-01.

[Rab83] Michael O Rabin. Randomized byzantine generals. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 403–409. IEEE, 1983.

[Rei96] Michael K Reiter. A secure group membership protocol. volume 22, page 31, 1996.

[Ric93] Aleta M Ricciardi. *The group membership problem in asynchronous systems*. PhD thesis, Cornell University, 1993.

[Ros14] M. Rosenfeld. Analysis of hashrate-based double spending. http://arxiv.org/abs/1402.2009, 2014. Accessed: 2016-03-09.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996.

[Sch90] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. volume 22, pages 299–319. ACM, 1990.

[SLZ16] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. Accessed: 2017-02-20.

[SSZ15] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. http://arxiv.org/pdf/1507.06183.pdf, 2015. Accessed: 2016-08-22.

[SW16] David Stolz and Roger Wattenhofer. Byzantine agreement with median validity. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 46. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[Swa15] Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf, Apr 2015. Accessed: 2017-10-03.

[SZ13] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains, 2013.

[SZ16] Yonatan Sompolinsky and Aviv Zohar. Bitcoin's security model revisited. http://arxiv.org/pdf/1605.09193, 2016. Accessed: 2016-07-04.

[Sza14] Nick Szabo. The dawn of trustworthy computing. http://unenumerated.blogspot.co.at/2014/12/the-dawn-of-trustworthy-computing.html, 2014. Accessed: 2017-12-01.

[TS16] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. In *IEEE Communications Surveys Tutorials*, volume PP, pages 1–1, 2016.

[VCB$^+$13] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, Lau Cheuk Lung, and Paulo Verissimo. Efficient byzantine fault-tolerance. volume 62, pages 16–30. IEEE, 2013.

[Ver03] Paulo Veríssimo. Uncertainty and predictability: Can they be reconciled? In *Future Directions in Distributed Computing*, pages 108–113. Springer, 2003.

[Vuk15] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[Vuk16] Marko Vukolic. Eventually returning to strong consistency. https://pdfs.semanticscholar.org/a6a1/b70305b27c556aac779fb65429db9c2e1ef2.pdf, 2016. Accessed: 2016-08-10.

[XWS$^+$17] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *Software Architecture (ICSA), 2017 IEEE International Conference on*, pages 243–252. IEEE, 2017.

[YHKC$^+$16] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where is current research on blockchain technology? – a systematic review. volume 11, page e0163477. Public Library of Science, 2016.

[ZP17] Ren Zhang and Bart Preneel. On the necessity of a prescribed block validity consensus: Analyzing bitcoin unlimited mining protocol. http://eprint.iacr.org/2017/686, 2017. Accessed: 2017-07-20.