Embedding reverse links in a blockchain

A. W. Roscoe
The Blockhouse Technology Limited
Department of Computer Science,
University of Oxford
University College Oxford Blockchain
Research Centre
Oxford, UK
awroscoe@gmail.com

Pedro Antonino
The Blockhouse Technology Limited
Oxford, UK
pedro@tbtl.com

Jonathan Lawrence
The Blockhouse Technology Limited
Oxford, UK
jonathan@tbtl.com

ABSTRACT

Blockchains provide extremely simple certainty looking backward in time because of the way each block contains a hash of its predecessor. This is not possible in the same way looking forward in time for various reasons, not the least of which is that when block N is created, block $N + 1$, and thus its hash, are unknown. Therefore blockchains rely on more complex mechanisms to establish what the successor of any given block is, and to ensure that alternatives - known as forks - cannot be introduced either close to the time of its creation or long after. These typically rely on chains of dependency and PKIs. In this paper we show how the concept of *hooks* can create something closely analogous to the usual hash links, only in the other direction. These represent a powerful mechanism to counteract attempts to insert forks from relatively old blocks, and are entirely internal to the blockchain. In understanding hooks we do some detailed combinatorial analysis of the game that good and bad agents play in blockchains, introducing criteria for relatively small collections of agents to make decisions, up to stochastic certainty.

KEYWORDS

blockchain, consensus, finality

1 INTRODUCTION

The basic data structure implemented by the backward hash pointers of blockchains is that of rooted trees. From every block there is a unique path back to the genesis block, determined by following the hash links. This is an easy process which we might call "heads-down": a process which can be performed by an agent knowing only a subset of global data and which, if it can be completed, is guaranteed to produce the correct result; as long as there is an index to look up blocks from their hashes there is no need to search for anything. This process is perfectly secure, given the usual assumptions about cryptographic hashes, because even if someone were to build a fraudulent index this would be easily detectable simply by checking the hashes.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Of course, once the blockchain is established and immutable up to block B, an index could be created mapping the hash of each block to the hash of its successor, or to the successor itself. But this would not be secure: as many alternative histories can be created as the enemy pleases, all rooted at the same block. The basic structure of a blockchain contains no evidence of which is the right one.

The context of a particular blockchain might well — and indeed should — support the correct successor. For example the Proof-of-Work (PoW) mechanism has the back-the-longest-chain rule and others seek endorsements for the correct successors. But these are not intrinsic to the blockchain structure, and require a "heads-up" (a process which requires an agent to actively gather as much data as possible from all sources, and even if it can be completed, may be incorrect because of some inaccessible data) search for the longest chain. The latter is potentially costly and — in the context where communication may be poor — not necessarily accurate.

In this paper we propose hooks: a special *hook transaction* $(hook, B, B', sig)$ is placed in the chain by (the creator of) block B that endorses a later block — the one agreed to be block B' — in a way that requires no reference to anything outside the chain, or even that creator's identity. After it is complete, it represents an endorsement of B' by B. The natural way to achieve this is for B to contain a one-time public key for signature, and rely on its private counterpart to create a signature of B' using it; given by the element sigin the hook transaction.

It requires a little more indirection than the usual hash pointers, in that the method relies on the creator of a later block, B'' to include this transaction. Our design relies on a parameter $r > 0$ to introduce some degree of redundancy on the creation of hooks. It dictates the number of agents that are required to be involved in hooking a particular final block. So, it overcomes any failure to perform on the part of agents who are involved.

Thus hooks, coupled with a block index, allow secure heads-down following of the blockchain upwards as well as downwards, up to the last hooks in the chain.

Hooks are thus an effective way to record consensus, and in particular finality that has been established in other ways, neatly into the chain itself, thus making it permanent and self-contained. They are no direct help in forming the consensus in the first place.

In the next section we introduce hooks and the mechanisms required to use them. We then give a formal definition of the structure of our model of hooking in the context of a blockchain. Finally we analyse the statistical model that underpins the trust in them.

2 BACKGROUND

Blockchains were initially proposed as a decentralised way to manage digital currencies and prevent double spending, i.e. the possibility of the owner of a digital currency from spending it more than once [10]. However, they have evolved into generic decentralised auditing systems that prevent much more than just double spending. For instance, with the advent of smart contracts — i.e. programs that are executed in the context of a blockchain — a developer can define by means of a program how transactions addressed to that smart contract are to be processed [2].

A blockchain is a *decentralised stateful transaction processing system*, sometimes referred to also as a *distributed ledger*. It receives transactions from its stakeholders, decides on which of those are valid, and perform alterations on its state that record the effects of these transactions. As a decentralised system, multiple agents collaborate to implement this behaviour.

A blockchain orders and stores valid transactions into *blocks* which are themselves ordered, giving rise, ultimately, to a *chain of blocks* representing the history of the blockchain. In practice, however, during its operation, a blockchain — or rather its agents — manipulate a *block tree*.

Definition 2.1. A block tree is a directed, finite and acyclic rooted tree defined by a pair (V, E) where V is a set of blocks and E is a set of backward links — the root is the only block without an outgoing edge. A block is a triple (id, hd, sig, bd) where id is the unique identifier of the block, the header hd of the block contains control information about the block, sig gives the header's signature by the block producer, and the body bd contains the list of transactions associated with the block. The header includes: the identity of the block producer $prod$, a cryptographic fingerprint of the block's body $hash_{bd}$, and the hash of its parent in the block tree $hash_{pt}$. We use the name of block elements as accessor functions in this paper. So, for instance, for block B, we can access the header using $hd(B)$, the producer using $prod(B)$, and so on. A backward link exists from block B_1 to block B_2 *iff* $hash_{pt}(B_1) = hash(B_2)$, where $hash$ is the cryptographic hash function used by the blockchain.

In this paper, we only consider block trees which are valid.

Definition 2.2. A block tree is *valid* if all its chains are valid. A chain is the path from a block in the block tree to the root. A chain is valid iff:

- Blocks signatures are valid — block producers are associated with a public signing key which allows participants to validate their signature on blocks;
- For each block B in the chain, $hash_{bd}(B) = hash(bd(B))$;
- Blocks transactions are valid.

Note that uniqueness of block identifiers and validity of backward links are already embedded into our block tree definition.

A blockchain system has ultimately to decide what is the one "true" chain within this block tree [5]. It typically does so by relying on two mechanisms. A chain selection mechanism by which, amongst valid chains/branches in the block tree, one is given priority. This is especially relevant in the context of block production. Producers will tend to use such a mechanism in deciding which chain to extend. The other mechanism is a finality mechanism [1, 3, 8, 12]. It decides when a block (and consequently a chain) becomes *final*/*immutable*. Normally, a blockchain system operates on a block tree with a few candidate chains springing off a common final/immutable block — before this block lies the immutable agreed-upon chain. In a well-behaved blockchain, the finality mechanism keeps extending the immutable chain by periodically deciding on a later final block amongst those candidate (mutable) chains, which results on the pruning of the alternative candidate blocks up to the height of this newly immutable block.

Our formalisation assumes the existence of an optional block header element that signals that a predecessor has been finalised. Such an element could also include the hash of a finality proof kept off-chain to support it. This proof is irrelevant for our exposition, we can just assume that good agents would be able to judge whether such an element is correct based on it. Moreover, the finality mechanism must prevent conflicting finality proofs (and block header elements) that would, for instance, show that two distinct blocks in different chains are final.

Definition 2.3. The block header element *final* is either *None* signaling that the element is empty, or it is $Some(H)$ signaling that the block with hash H is final. A validity requirement for this element ensures that the block with hash H is either the block with $final = Some(H)$, or one of its predecessors.

3 RELATED WORK

The concept of *hooking* in blockchains was first proposed in 2020 by Roscoe and Lei [11], and the present paper is a development and formalisation of ideas from that document.

Hooks are used to record and harden finality, so it is relevant to consider previous work relating to finality in blockchains. Finalisation of a block that has been appended to a blockchain refers to the point at which it becomes irrevocable. There are several variants on finality characteristics, for example deterministic or probabilistic, immediate or eventual.

Anceaume et al. [1] study several deterministic finality properties and investigate their provability. This is achieved by introducing the notion of bounded revocation, which is based on the assertion that the number of blocks that can be revoked from the current head of a blockchain is bounded. Using this revocation number as a metric, results are obtained relating (by reduction) different forms of eventual finality, and including impossibility results.

In Stewart et al. [12], the authors introduce the concept of the finality gadget. A finality gadget allows for transactions to commit optimistically, but clients must assume that these transactions might be revocable until the gadget confirms their finalisation. As a result, a blockchain can execute transactions optimistically and only commit them after they have been sufficiently and provably audited by the gadget. The paper includes a formal model of the finality gadget abstraction, and uses this to prove several results including impossibility results about the model.

We anticipate that hooks are most likely to be applicable in proof-of-stake based blockchains (because of the ease with which forks might otherwise be forgeable). Buterin et al. [3] introduces Casper, a proof-of-stake based finality gadget which augments an existing proof-of-work blockchain. The idea is that the finality gadget can be "run" alongside the existing blockchain as it is extended, certifying blocks as final once they are irrevocable according to its criteria.

The authors claim that Casper can provide almost any proof-of-work chain with additional protection against block reversions. Hooks would provide an ideal mechanism to record the results of a finality gadget back into the target blockchain.

Another paper considering finality in the context of a proof-of-stake blockchain is [8]. This describes the "Ouroboros" blockchain protocol, claiming to be the first such protocol based on proof-of-stake with rigorous security guarantees. The paper derives security properties for the protocol comparable with those achieved by the Bitcoin blockchain protocol.

A characteristic of hooks is that they can potentially be used with any finality mechanism, although those providing deterministic and bounded finality are most obviously suitable. For probabilistic finality, there would be a threshold minimum probability level that a block is final, at which a hook should be placed. This will obviously be very close to 1. For eventual finality, it would be necessary for the originator of the hook to remain active until the block being hooked is deemed final.

4 WHAT IS A HOOK?

Hooks are proposed as an alternative mechanism to *record finality* in the blockchain. They are meant to either provide a concise and consensus-protocol-independent alternative to convoluted finality proofs, or to provide extra security guarantees to weak finality proofs. For instance, if finality proofs do not use quantum-resistant cryptography, one could use quantum-resistant hooks to create a quantum-resistant records of finality. The ambition should be that for an adequately hooked section of blockchain, there is no need to remember the proofs of finality that the hooks summarise.

A hook is created by an entity that is called a *picket*. For each block being produced, a corresponding picket is chosen independently and at random from a set of eligible pickets. To simplify our exposition, we make pickets and block producers coincide, and assume that block producers are chosen accordingly.

A hook involves three blocks: *base*, *tip*, and *arc*. The base block is where the public key, later used to verify the hook, is published, the tip correspond to a final block that needs hooking, and the arc is the block in which the hook transaction, defined next, is published.

Definition 4.1. A blockchain can be extended as follows to accommodate hooks:

- A block header contains an additional $hook_{pub}$ element which is a strong and quantum-resistant public signing key — the corresponding secret key is solely used to create hooks;
- It accepts an especial hook transactions. A hook transaction is a triple (*base*, *tip*, *sig*) where *base* and *tip* are identifiers for the corresponding blocks, and *sig* is the signature of block *tip* using the private key dual to the public one published in *base*.

We assume that the blockchain relies on a well-known cryptographic signature scheme which is used to generate strong keys, and create and verify signatures. The signing scheme for the hooks transactions must be additionally quantum resistant.

A hook transaction is valid if *base* and *tip* blocks exist in the chain containing this transaction, and the *sig* is a valid signature of *tip* using the private key corresponding to $hook_{pub}(base)$. The existence of duplicate hook transactions in the same chain is forbidden.

All an agent's hooks are linearly related: they link blocks in one linear chain. Thus, for example, no agent may generate more than one hook per block. We believe that a good option is to place a public key for one-time signature in each block, most probably one based on a hash-based signature scheme, and to use it exactly once. Such schemes are believed to have very strong quantum resistance provided the hashes involved are wide enough.

4.0.1 Hook creation. Before we introduce how the hook creation mechanism works, we introduce some relevant terminology. We call a block F for which there is a valid pointer $final = Some(id(F))$, and consequently a finality proof, a *final* block; the block P containing this pointer is the *prover* of F. Note that P and F could be the same block. The final block and all its predecessors are *finalised* blocks but only the ones with such pointers/proofs are called *final*. We say that block B has been finalised by block F iff F is final and either $F = B$ or F is the closest final successor of B — or, equivalently, we say that F finalises B. Note that B, F, and P could all be the same block — this would be the case in blockchains using instant finality protocols. Moreover, we use $B(i)$ to denote the i^{th} block ahead of B in the true chain; $B(0) = B$, $B(1)$ is the next block forward and so on. We also use $B(-1)$ to denote the predecessor of B and so on. We use $F(B)$ to denote the block that finalises B, and $P(F)$ to denote the prover of F.

In our hooks mechanism, B's picket produces a hook transaction with block $F(B(r))$ as the hook's tip, once it has seen the block $P(B(r))$, namely, it waits for the r^{th} block beyond B to be produced and creates a hook targeting the first final block in the interval , once it is known — $r > 0$ is a parameter of our hooking mechanism that we discuss later. In order to receive block rewards and/or its deposit back, a block producer is obliged to create the corresponding hook.

Anyone can judge the validity of a hook, and the party generating it should post it in place(s) that should be universally seen. The consensus/validity process can then ensure that where there is a hook available it is placed in a block as soon as possible.

Note that a blockchain using instant finality and $r = 3$ is expected to exhibit a hooks structure similar to the one presented in Figure 1. However, the ones using eventual finality would be expected to, instead, depict collections of hooks targeting the same block — intuitively, we say that a block B hooks a block B' iff a valid hook transaction is published with base B and tip B'.

4.0.2 Hooks verification. Once hooks are placed, they can be used to distinguish the true chain from forking attempts. This distinction is made based on the number of valid hooks placed in a given chain. Intuitively, we expect a certain number of hooks targeting a final block to make it *anchored*.

An agent can use the following hooks verification protocol to check whether a given chain is the true chain. It starts on the genesis block and moves forwards on the chain. In this forward traversal, each final block, in turn, is tested for *anchoredness*. The traversal stops as soon as a final block is found not to be anchored. We call this verification algorithm *verify_hooks*. Thus, the hooks verification can only prove the finality/immutability of the chain

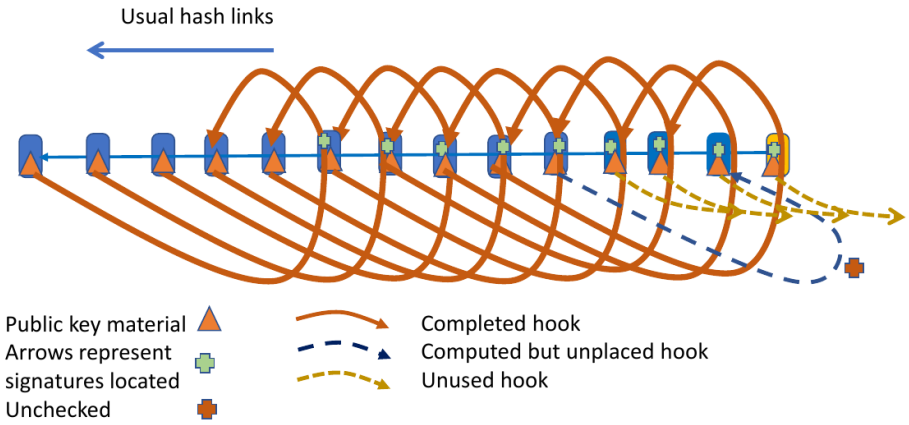

Figure 1: Idealised structure of hooking within the true chain

up to the last anchored block. For final block B, anchoredness is tested using the function *anchored*(B), which we describe next.

Let Δ be a parameter of the hooking verification mechanism corresponding to acceptable time to have a hook transaction created and published. Let F be a final block and F' the final block immediately preceding it. Intuitively speaking, block F is anchored by the $r > 0$ blocks ranging from $F'(-r+1)$ until F', and we analyse how many of the hooks having one of these blocks as a base have been corrected placed/published by analysing the corresponding hooks transactions present in blocks $P(F)$ to $P(F(F'(r))) + \Delta$. Note that the hooks created by block producers in the interval $F'(-r+1)$ until F' target the final block F or even a final block that succeeds F, if F precedes $F'(r)$, given the rules that we outlined for the placement of hooks. So, we choose these pickets to be the validators of (i.e. to anchor) F. Furthermore, the interval between blocks $P(F)$ and $P(F(F'(r))) + \Delta$ constitutes the acceptable period under which the corresponding hooks must be published/place. Hence, the focus of our verification procedure in this specific publication interval.

Definition 4.2. The block F is anchored iff there is at least *accept*-many hooks transactions in the interval of blocks from $P(F)$ to $P(F(F'(r))) + \Delta$ with hook bases in the interval $F'(-r+1)$ and F'.

We propose a framework that requires at least g-many good agents to have contributed to the anchoring of a block and we want to tolerate at most u unused hooks in the window of hooks transaction that we analyse for anchoredness. Thus, we need to ensure that at least $k = g + u$ good agents are amongst the r agents selected to anchor block F, so that $accept = r - u$. Note that the value u gives our framework some flexibility. Instead of requiring the publication of exactly r hooks, it can accept the lower threshold of $r - u$.

We use a stochastic analysis that relies on the notion of the *goodness lower bound* to estimate r. This notion provides a stochastic lower bound to the number of good agents in r consecutive block producer selection. We assume that block producers are selected independently and with probability p of being good. Moreover, our analysis relies on a stochastic insignificance threshold e: if the probability of an event is below e, we consider it impossible to happen in practice.

Definition 4.3. The *goodness lower bound* is given by $GLB(N, p, e) = l + 1$ where l is the largest l such that $F(l; N, p) \leq e$. If no such l exists, $GLB(N, p, e) = 0$. More formally, in N consecutive producer selections, the probability of having at least $GLB(N, p, e)$ good agents is at least $1 - e$.

- $F(l; N, p) = \sum_{i=0}^{l} \binom{n}{l} p^i (1-p)^{N-i}$ is the cumulative distribution function for the binomial distribution where N is the number of trial, l the number of successes, and p the probability of a success.

So, we use the smallest r such that $g + u = GLB(r, p, e)$. To illustrate these calculations, if we had $e = 10^{-18}$ and $p = 0.67$, we would have the following *GLB*s: for $r = 20$, *GLB* is undefined; for $r = 40$, $GLB = 1$; for $r = 60$, $GLB = 8$; for $r = 80$, $GLB = 16$; for $r = 100$, $GLB = 24$. If we further define $g = 5$, we have: for $u = 10$, $r = 78$; for $u = 20$, $r = 101$; for $u = 30$, $r = 123$; and for $u = 40$, $r = 143$. The parameter p is chosen based on the fact that consensus protocols with byzantine agreement [4, 9] typically require that more than 2/3 of the participating agents be good [6, 7], whereas $e = 10^{-18}$ is a negligible stochastic insignificance threshold.

The only downsides to this hooking mechanism are: (i) that the agents are obliged to pay attention to the evolving consensus for longer to enable the hook to be placed correctly — they need to wait for at least r blocks to then place their hooks — and (ii) we need to wait longer before relying on hooks to preserve the chain.

5 SECURITY ANALYSIS

The main point of hooks is to prevent the following type of attack:

Suppose block M is a legitimate element of a chain C — with legitimate successor M' — that is well behind the current head. Thus, it is not gaining attention from the consensus algorithm. The blockchain does not rely on a PoW-consensus and consequently it is possible for an enemy to create a long chain in a relatively short time.

The enemy, acting as agent E, has had a plausible excuse to create the successor to M. Perhaps E did create the true successor in C, or was elected to do so and did not. Perhaps some of identities involved in creating earlier blocks have disappeared or their cryptography compromised. E creates an alternate successor $M^{\square}$ and enlists other bad agents to create a chain C' of successors to that, which contain supposed facts that will help to defraud some agents who might not yet be fully engaged with C.

When such an agent A wishes to use C, the enemy tells it that the head is the head of C'. If A believes this, she can be defrauded. If A were told of C' in a blockchain with hooks, there would be no hooks present to back $M^{\square}$ from any of M and its predecessors that were created by good agents. What we therefore need to ensure is that everyone knows — correctly — that sufficient hooks from good agents are present backing every block of C, and furthermore be in a position to tell when this fails.

We work on the following assumptions:

Block creators are chosen independently at random from a pool of willing parties, and the probability of one being chosen that follows all the rules is more than 2/3 because that is the lower bound for Byzantine agreement to be correct [6, 7].

A good party will always contribute what is expected of it when it is participating. In particular any agent that has contributed a block but has not yet contributed the corresponding hook will be aware of all blocks upon which consensus has been agreed that are later than its own block.

Failure to produce a hook on time means that the delinquent agent is still expected to produce one before it is allowed to do anything else and another formulaically chosen agent before those already involved is invited to produce a substitute that is labelled as such. Thus if the delinquent agent permanently fails to come up with the goods it is almost surely bad.

However whatever mechanism is available to the good agents to repair any pattern of hooks when bad ones do not offer theirs and thus retire from the chain is also available to bad ones repairing the holes in a fork. Consequently we need to put in place a mechanism the opponent cannot replicate, so that it does not even try. We will discuss this below.

It follows that any block that has existed in the chain for some time will have r, or at least very nearly so, confirmations from blocks below it that it, or some block with it as a predecessor, lies on the true chain. Anyone should believe those that come from good agents, but be skeptical of those from bad ones. The only problem is that they do not know which is which.

The twin dangers we have to face are that bad agents will not place the hooks they are supposed to or will place them to back a fork — whether or not they have not backed the true chain. What we need to do is ensure that when reviewing a block to see if it is on a fork, every block is backed by a pattern of hooks that it is impossible for the opponent to achieve. Therefore we need to explore the limits of what the opponent might achieve. This is of course dependent on the distribution of bad agents amongst block creators.

We assume that, whatever way block creators are chosen, the distribution is that of a biassed coin with some probability $p < 1/3$ (as discussed earlier) of picking "bad". We assume that the identities of the bad agents are known to each other but not to anyone else until one commits an act in public that proves it bad.

Just as a coin toss can produce an arbitrarily long run of heads, our selection can produce an arbitrarily long run of bad agents. But at some point this becomes so unlikely that we can safely ignore it — in the real world this will not happen. Since the number of bad agents picked in a run of N is a binomial distribution with standard deviation $\sigma = \sqrt{p(1-p)N}$ and mean $\mu = pN$ where p is the likelihood of one selection being bad, lay down an extreme criterion for stochastic certainty by assuming that the distribution will never fall outside $k\sigma$ for a suitable k. In other words, we are stochastically certain that there will be no more than (i) $k\sigma + \mu$ bad agents.

We pick 10^{-18} or conservatively $k = 9$ as that corresponds to an unlikelihood of about 0.11×10^{-18}, namely, the probability of having more than $k\sigma + \mu$ bad agents amongst N trials. If a billion (10^9) trials are made, this unlikelihood will still almost surely never happen. We assume that p is less than 1/3 as that is the limit for Byzantine agreement working. By applying these values to (i), we have stochastic certainty of there being no more than $N/3 + 3\sqrt{2N}$. Thus, for $N = 60$ we can guarantee that there are at least 7 of any 60 consecutive blocks that have good agents nominated to create the hooks. Of course we expect a lot more. The larger N is, the greater the proportion of the N can be guaranteed to be good: on the same basis $N = 44$ is the smallest number to clearly guarantee that one is good, whereas $N = 100$ would guarantee 36 good. Note that however large r is, the basic protocol only expects each block creator to create a single hook. Of course, if we can guarantee that p is less than some number less than 1/3, these lower bounds on the number of good nodes will improve further.

Assume that likelihood of a bad miner is 1/3 as that is the upper bound of Byzantine agreement. Thus N trials gives variance $2N/9$ so standard deviation is $0.471\sqrt{N}$. So $9\sigma = 32.9$ for $N = 60$. It is "certain" that if there are more than $N/3 + 9s$ hooks backing a particular block then it is correct. We can conclude certainty for 53 or more in that case.

However:

- To be stochastically certain at 10^{-18} that there are less than half bad we need N at least 614 when $p = 1/3$. That is a telling number, since if the process for eliminating branches never stretches that far back, a determined attempt to back a fork might just win.
- Within a run of 60 consecutive trials, the likelihood of at least 30 of them being bad is about 1.5%, whereas the probability that 7 or less are bad is less than that — thus to get 53 hooks we probably need bad agents to contribute.

This emphasises that in order to get a hook length of 60 to work reliably you almost always have to rely on bad agents placing hooks correctly if there really are nearly 1/3 bad agents. In order to achieve this the chain applies penalties, such as making the placing of a correct hook compulsory before doing anything else, and reputational and financial damage. It also needs to make the opponent aware that if they were not to participate apparently properly standard behaviour, they would face a back-up mechanism that would defeat him.

That is true across other fields related to blockchain consensus, and relates strongly to our proposed mechanism for consensus itself. The key issue both there and here is the need — up to stochastic certainty — to distinguish between the correct appearance of a good and proper chain, and the appearance of attempts (like forking) to subvert it even though bad agents can contribute malignly or not at all. In considering this we have to distinguish between the effects the opponent can have on the true chain, perhaps by non-participation, and the story it can stitch together, always bearing in mind that the observer does not know a priori which is which.

Inspired by the above analysis of hook patterns and standard Byzantine agreement, we introduce two general criteria for making decisions in blockchains and similar decentralised systems with an assumed probability p of any agent being bad and an established criterion for stochastic impossibility. In each case every one of N agents is asked to make a decision from a set D. The assumption is that a bad agent may do just this, or may have the following choices dependent on circumstances:

- Independently tell other agents (both inside and outside the set of N) anything or nothing.
- Make a choice of what to say and say that to everyone.

We assume that any good agent will make a correct decision that everyone will be happy to accept, and that we want all good observers and participants to draw the same conclusion after seeing the verdicts.

In any case where it is known that all good agents will agree, an observer can accept a dominated decision: sufficiently many verdicts from the set that it is stochastically impossible that they are all from bad agents.

In any case where no agent can tell different things to different observers, we can accept a dominating majority: a majority of the N agree on the decision and the number who agree is greater than the maximal number of bad participants in the N. Someone seeing this will have a clear indication of the decision — there can only be one majority — and that it is an acceptable one.

In the general case, to make a decision, we require a dominating strong majority in favour of an outcome. This means that so many agents back a decision that it is known that a majority of the good agents do. In general if a dominating majority of N is (minimally) k this means that at least $N - k + 1$ are good and at most $N - k$ of the good agents do not agree with the majority. Consequently we can enumerate the corresponding dominating strong majority as $(N + k)/2$ agreeing. We can be sure that if two observers both see a dominating strong majority then they see the same value as the result. We cannot be sure that all will see a dominating strong majority.

However if all the verdicts that an agent A sees are signed by the originators and these agree on the provenance (i.e. the basis of the decision), then A can collect together these verdicts into a certificate for the decision. Anyone seeing this certificate will know the decision is both unique and correct.

5.1 Formalising the safety of hooks

Based upon the same flavour of stochastic analysis, we formulate an inductive argument to demonstrate that hooks can be used to correctly identify the true chain of the blockchain. We rely on two assumptions about the behaviour of the underlying blockchain.

- *Producer Selection Assumption* (*PSA*): if B is a true final block then all producers for B and preceding blocks must have been selected independently and with probability p of being good.
- *Hook Validity Assumption* (*HVA*): if g-many good agents have hooked a block, the block must be a true final one.

Claim 5.0.1. *Let B' be a true final block and B the next final block in the chain. If B is anchored then B is (stochastically certainly) a true final block (with probability $1 - e$).*

Proof sketch. The proof is based again on our stochastic analysis. Given that B' is a true final block and by *PSA*, we can deduce that the r agents selected to produce blocks $B'(-r + 1)$ to B' were selected with probability p of being good. It follows from our definition of *GLB*, *accept*, and *anchored*, then, that at least g good agents have contributed to the hooks for block B with probability $1 - e$. Therefore, by *HVA*, we can deduce that B is a true final block with probability $1 - e$. □

By induction on the sequence of final blocks, one can derive the following corollary about our *verify_hooks* algorithm. The starting block in this sequence, however, must be assumed/agreed to be final — *a hooks starting checkpoint* — and must have $r - 1$-many predecessor blocks so that anchoredness can be checked for the next final block in this sequence.

Corollary 5.1. *The verify_hooks algorithm (stochastically certainly) identifies the true chain up to the last anchored block verified.*

6 CONCLUSIONS

We have described the concept of hooks, an addition to the blockchain data structure that ensures that forks cannot be introduced after the chain has developed. These are succinct, and independent of the consensus algorithms being used.

Hooks is a mechanism that strengthens or simplifies the recording of finality. So, for hooks to add value to a blockchain's behaviour they must either provide stronger cryptography/security or be simpler than the original finality proofs. Hooks could, for instance, provide quantum-resistance for a finality mechanism using proofs that are not quantum resistant.

In understanding the combinatorics of the games that good and bad agents might play using hooks, we have developed criteria for de facto certainty to be derived from them, and come to understand how it is essential for opponents to know that any conceivable attack based on non-participation can be countered. This means that we can adopt an efficient and rapid hooking protocol.

In future work we will show how similar analysis can lead to secure and efficient consensus algorithms.

ACKNOWLEDGMENTS

We thank Professor Wang Lei for challenging Bill Roscoe to find an elegant solution to the problem of preventing fork formation in blockchains. The concept of hooks arose directly from this.

REFERENCES

[1] Emmanuelle Anceaume, Antonella Del Pozzo, Thibault Rieutord, and Sara Tucci-Piergiovanni. 2022. On Finality in Blockchains. In *25th International Conference on Principles of Distributed Systems (OPODIS 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 217)*, Quentin Bramas, Vincent Gramoli, and Alessia Milani (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 6:1–6:19. https://doi.org/10.4230/LIPIcs.OPODIS.2021.6

[2] Vitalik Buterin. 2014. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. https://ethereum.org/whitepaper/.

[3] Vitalik Buterin and Virgil Griffith. 2017. Casper the Friendly Finality Gadget. *CoRR* abs/1710.09437 (2017). arXiv:1710.09437 http://arxiv.org/abs/1710.09437

[4] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (New Orleans, Louisiana, USA) *(OSDI '99)*. USENIX Association, USA, 173–186.

[5] Benjamin Y Chan and Elaine Shi. 2020. Streamlet: Textbook Streamlined Blockchains. Cryptology ePrint Archive, Paper 2020/088. https://eprint.iacr.org/2020/088 https://eprint.iacr.org/2020/088.

[6] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)* 35, 2 (1988), 288–323.

[7] Michael J Fischer, Nancy A Lynch, and Michael Merritt. 1986. Easy impossibility proofs for distributed consensus problems. *Distributed Computing* 1, 1 (1986), 26–39.

[8] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology – CRYPTO 2017*, Jonathan Katz and Hovav Shacham (Eds.). Springer International Publishing, Cham, 357–388.

[9] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (jul 1982), 382–401. https://doi.org/10.1145/357172.357176

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

[11] A W Roscoe and Wang Lei. 2020. Taking the work out of blockchain security. https://tbtl.com/wp-content/uploads/2020/09/Taking-the-work-out-of-blockchain-security.pdf.

[12] Alistair Stewart and Eleftherios Kokoris-Kogia. 2020. GRANDPA: a Byzantine Finality Gadget. *CoRR* abs/2007.01560 (2020). arXiv:2007.01560 https://arxiv.org/abs/2007.01560