Incentive-based interactive settlement scheme for computational integrity arguments (IBISS-CIA)

Maxim Orlovsky[1,2] and Sabina Sachtachtinskagia[1,3]

[1]Pandora Core AG, Switzerland
[2]BICA Labs, Switzerland
[3]Athens University of Economics and Business, Greece

***Abstract*—Censorship resistant decentralised computing is a new topic in distributed systems theory. With the recent rise in popularity of Bitcoin as a censorship-resistant financial and transactional system a new technological field of blockchain and its applications appeared. This work addresses the problem of computational integrity proofs (correctness of the actual computations) made by some non-trusted (anonymous) third party without repeating the whole volume of the actual computing. The issue is usually addressed by use of probabilistically checkable proofs (PCP), however, such simple approach leave untouched the question of coupling the payments for the consumed computational resources with the actual proof. The purpose of our protocol is to scale distributed calculations by obtaining results of nearly perfect (measurable) quality, without the need to repeat calculations or check the computational integrity proofs by the client. This is achieved by providing meritocratic economic incentives to all participating economic agents, so that their interests are aligned with the network's success. As a result, the need for repeat calculations diminishes, and costs of calculations become lower and competitive. Ability to run PCP schemes without trusted checking party provides a way to codify the actual protocol in terms of smart contracts, so it can be used in a very compact form with existing economic blockchain layer. Our approach to protocol design aims at the smallest possible size of the shared system state so it can be implemented with any non-Turing complete smart contract system with storing nothing else then the account balances for the involved parties. For example, the protocol can be implemented with the existing Bitcoin script functionality with only two on-chain transactions per whole protocol computing cycle. In such system the PCP proofs are run the second layer on top of actual Bitcoin blockchain, reducing the footprint, price and increasing scalability for the real-world business cases for the computing.**

Keywords*—computational integrity, probibalistically checkable proofs, censorship resistance, decentralisation, distributed systems, Byzantine fault tolerance, game theory, cryptoeconomics, economic incentives, parallel computing, zero-knowledge computing

I. Introduction

The progress in the field of computing led to the appearance of cloud computing platforms. The success of cloud computing was related to the economy of scale phenomenon, enhanced by the exponentially increasing amounts of produced data [27]. However, it has created systematic risks and fragility, namely privacy risks, possibilities of totalitarian control, surveillance, single points of failure and possibilities of censorship [18], [25]. It has put at risk the further development of machine learning technology, started to be monopolised together with the access to big data. The #FreeAI Manifesto [3] had declared an initiative to oppose this dramatic trend, and this work aims at the development of the technology stack, which can address the issue.

The problem of censorship-resistant computing can be split into the following parts:

1) Finding and buying information (big data, machine learning models) in a censorship-resistant way.
2) Owning and keeping information (big data, machine learning models) private (with possible plausible deniability).
3) Performing high-load computing on big data in anonymous and censorship-resistant and (desirable) private manner (for the cases when personal computing resources are not enough to perform the actual computation).

In the present work we address only the third matter. The first two matters can be addresses with many other existing initiatives, such as dark anonymous markets with escrow, cryptography technologies, zero-knowledge, distributed storage and data transfer (BitTorrent, IPFS etc) [8], [11], [16], [12], [9]. We attempt to develop a model that would estimate the possible severity of the problem and provide a solution. We use Games Theory methods to study the incentives of network participants and propose ways to align their incentives with those of the network as a whole. The network is therefore redesigned in such a way that all participants, following their "egoistic" incentives and maximizing their own profits, arrive to contributing to network's success.

The technical problem we need to address here is how to prove computational integrity (correctness of the actual computations) made by some non-trusted (anonymous) third party without repeating the whole volume of the actual computing. This is done by enhancing probabilistically checkable proofs (PCP) [7], [14] with the ability to run PCP schemes without trusted checking party. Such protocol can be codified in terms of smart contracts, so it can be used in a very compact form with existing economic blockchain layer. Our approach to protocol design aims at the smallest possible size of the shared system state so it can be implemented with any non-Turing complete smart contract system with storing nothing else then the account balances for the involved parties. For example, the protocol can be implemented not only with WASM [2] or EVM [26] systems, but also using the existing Bitcoin script functionality with only two on-chain transactions per whole protocol computing cycle. In such system the PCP proofs are run the second layer on top of actual Bitcoin blockchain (or it's sidechains [5], [23]), reducing the footprint, price and increasing scalability for the real-world business cases for the computing.

The proposed solution is twofold. Firstly, we introduce a more meritocratic approach, where network participants need to assume responsibility by putting something at stake if they want to receive gains from the network. The greater the rewards that a participant opts for, the greater is the stake that will be lost if the participant does not produce contributions of acceptable quality. Secondly, we develop a process of arbitration, which is essentially a process of quality control and network protection applied specifically to distributed ledgers. Arbitration is about re-checking random parts of work done by network participants and resolving conflicting opinions whenever they occur. We also develop instruments that help to estimate the probability and costs of attack on the network, whether for personal gain or for disruption of its services. Finally, we attempt to align the incentives of all network participants so that the network would be able to use quality control sparingly, e.g. checking only 10%-15% of all work done, and still achieve high quality results.

II. Previous work

The development of decentralized environments in general and smart contracts in particular is a very recent phenomenon, hence there is not much economic literature existent in this field. The generic source on which we build our study is the theory of incentives, and in particular the topics of moral hazard and adverse selection, e.g. as analyzed extensively in the seminal book by Laffont and Martimort [20]. However, that literature refers mostly to vertical chains, while our study pertains to multi-agent decentralized networks. Another stream of economic literature worthy of reference is that about the free-rider problem. For a description of this problem from game theory perspective, see e.g. chapter 6.2 in Fudenberg and Tirole [13]. In a sense, the nodes of blockchain that try to receive rewards without doing their share of work can be seen as free-riding on the network. Despite this similarity, the blockchain by definition may not have a central planner, but only a protocol that acts according to predetermined rules that were accepted voluntarily by all participants of the network.

At the same time, our work is closely related to the recent cross-disciplinary game theory and computer science research on proof of work, proof of stake and other types of consensuses. There is a significant literature on the proof of work protocol that allows to remove the implicit assumption of the honesty of the majority miners, and proves instead that their incentives to be honest by finding the Nash equilibrium that they follow. Some important contributions on this topics are [15], [6]. The specific topic of computations verification, which is the central part of our work, has been previously tackled e.g. [19], albeit in Bitcoin mining environment. On the other hand, the idea to use economic incentives through collateral stakes in order to secure the network is widespread nowadays, see for example [17], [10]. However, our model is distinctly different from these streams of literature because we use at the same time stakes incentives (punishment for cheating), work incentives (payment for the actual useful work done) and do not require a global consensus. Since our point of interest are calculations and not coin mining itself, we therefore avoid the discussion about consensus and need not examine here the coin mining incentives.

A model that indeed bears similarity to ours is found in [24] that describe the Truebit project. Their approach is different from ours primarily because they use a pool of validators that work on checking calculations simultaneously for a bounty. While adding to network's reliability, that raises the cost of verification to over 500% of the cost of the calculation itself according to the same authors estimation, which is not very competitive compared to the centralized solution. In contrast, our aim is to achieve reliable calculation results while checking only a small part of calculations by using economically incentivized, randomly selected validators.

Probabilistically checkable proofs (PCP) are well-known interactive and non-interactive schemes for computational integrity arguments [7], [14]. However, they leave untouched the question of coupling the payments for the consumed computational resources with the actual proof. The purpose of our protocol is to scale distributed calculations by obtaining results of nearly perfect (measurable) quality, without the need to repeat calculations or check the computational integrity proofs by the client. This is achieved by providing meritocratic economic incentives to all participating economic agents, so that their interests are aligned with the network's success [4], [22]. As a result, the need for repeat calculations diminishes, and costs of calculations become lower and competitive.

III. Game model

A. Initial setup

The protocol design includes the following types of participants:

Client node C**:** party prepared to pay for the actual computing and provide necessary data for it.
Worker node W**:** party prepared to perform actual computing for some reward s.
Verifier node V**:** party prepared to verify the actual computing using PCP for some reward $s/z \ll s$
Arbiter nodes A**:** special party required only in case of **Arbitration** procedure (see III-G below)

The client node splits the actual computing task into **batches** suitable for parallel processing in a distributed P2P network. The protocol run for each separate batch independently.

The Client comes with a take-it-or-leave-it offer that Worker and Verifier can agree to sign. The offer is a contract to do calculations (Work) for a Payment, which is split in a predetermined way between W and V. The Worker shall calculate all the Work, while the Verifier shall calculate only a small portion of Work that is called Sample (described more fully at III-C).

B. Selection of Worker and Verifier

The client selects Worker and Verifier nodes on some open market or by auction (the actual selection is not part of this protocol). They agree upon the price of the computing and stakes (see below) in the process of off-chain P2P communications, like ones that can be built on top of Kademilia [21] or libp2p

[1] protocols. The Client comes to independent agreements with the selected Worker and Verifier about contract terms. The contract for Work is signed by all three parties: C, W, V - this ends this stage. Until the contract is signed, there are no further stages.

According to the contract, the *Payment* (c), the *Worker Collateral Stake* (w) and the *Verifier Collateral Stake* (v) are deposited to *Escrow*, to be released under certain conditions described in this text. Since calculations incure costs (hardware, electricity), the incentive to report false results without doing actual calculations is countered by providing collateral stakes that can be slashed if cheating is detected. It is economically logical that v is larger or equal to w, where the case $w = v$ has certain merits of symmetry against attacks on stake. It is also economically logical that Verifier's work should be more expensive (per unit of calculations). This is because Verifier's stake is locked for approximately the same time as Worker, but Verifier's total reward is less than Worker's total reward (since much less work is done).

C. Worker's calculations of Work

The Worker proceeds to calculate the Work. Then, a part of ready Work is chosen as a Sample for verification purposes (according to a known tech process not described here). The Sample is chosen in a way that the Worker does not know which part of Work will be checked and therefore cannot choose which part of Work to calculate selectively. An economically feasible size of such Sample is between 5% and 20% of the Work. The Sample result is reported to the Client – this ends this stage.

If the Worker fails to send the Sample result by a certain time, then he is penalized (not necessarily with a full stake w), and a replacement Worker can take this contract. Alternatively, if that it easier, the contract could be dissolved and everyone except the Worker receives full refund of deposits, while the Worker is penalized (not necessarily with a full stake w). This penalty prevents the stalling attack. The money from penalty could be burnt or given to the Client (and/or Verifier, if he was already chosen) as a compensation for wasting time.

If the Worker sees that data are invalid and cannot be used for calculations, then there are various options to deal with the problem. One easy option is to allow the Worker to proceed as normal with the contract, and the Verifier and Arbiters should consider any results reported by the Worker as correct, and Client must give the full amount of Payment for this Work. This option has transparent design and should lead to a provable Nash equilibrium. Other options, in my opinion, are less solid and should be considered only if the first option is impractical for "marketing" reasons (it punishes Client's mistakes harshly).

The second option is to introduce a special subsequent stage which allows the Worker to back off from the contract after being the data (if this is allowed without penalties, it makes possible a stalling attack by the Worker, but if we add a small penalty, it makes possible a flooding attack by the Client, so neither solution is perfect). Another option is to have Worker and Verifier both agree on poor quality of data, in which case the Client can receive refund of Payment minus a small penalty (while if Verifier does not agree with Worker on poor quality of data, then either Arbitration, or the Worker can back off from the contract with some penalty). Whether the Client should be entitled to start Arbitration over the quality of data is questionable: it creates a trade-off between burdening the Arbiters with the trivial task of data checking (if C, W, V do not all agree) and assigning minor penalties to Client that cannot be protested in Arbitration III-G.

D. Verifier's calculations of Sample

The Verifier receives the information how to calculate the Sample. The Verifier calculates the Sample and reports to the Client. Hence, the Client gets the information whether the Worker's results are considered correct by the Verifier – this ends this stage. Note that it is not critical whether the Verifier received the Sample value calculated by the Worker. Both settings are possible:

(a) where the Verifier reports his own Sample result (without knowing the Worker's Sample result)
(b) where the Verifier reports only a binary agreement or disagreement with the Worker's Sample result

However, the setting (a) is better, as it makes collusion between W and V more costly.

If the Verifier fails to report to the Client by a certain time, then he is penalized (not necessarily with a full stake v, however it should be a significant penalty, not a minor one). This penalty prevents the stalling attack and other combined attacks. In this case, a replacement random Verifier needs to take over the contract, perhaps one that is chosen initially as a replacement. The money from the penalty could be burnt. If the Verifier's failure to report to the Client can be detected automatically then some of penalty the money could given to the Client and/or Worker as a compensation for wasting time. However, if the Verifier's failure to report to the Client is not automatic, but needs to be confirmed by the Client, then it wiser not to give to Client any monetary rewards for penalising the Verifier, and the Verifier should be able to appeal to Arbitrage.

More generally, whenever and party receives money extracted from the stake of another party, this creates incentives to attack by misreporting another party's failure. This is why any significant portions of stake taken from W or V should never be fully split between C, W, V - a large part of the penalty money should be burnt (unless used to pay for Arbitration), while minor compensations can be given. The stakes exist mostly to provide payment for Arbitration, if such a need arises. Minor compensations to C, W, V may be useful to offset attacks that waste time of network participants (I call these "stalling attacks").

If inserting a replacement Verifier into a contract is technically very hard, there is another possibility. The contract could be dissolved. In that case, the Worker is paid for the Work fully, by money from Verifier's penalty. The Client receives back his full Payment plus the Work results, unverified.

As a sidenote: a question may arise, if the Client, having received the Sample calculations by two parties (which in this case would be two Workers) can be entrusted with reporting whether the results agree. That would make the Verifier's role redundant. However, in my opinion, the one who produces the Sample calculations checking (whether we call him V or "another W") should also be the one who reports whether it agrees with the initial Worker's result. Or (even better) if this can be done automatically by comparing the two values. The Client, of course, can easily understand whether two values, reported to him, agree or not – however, preferably he should not be the one who reports this. Adding agents (including the Client) to the chain of information flow would complicate incentives and create "broken phone" situations, where information changes as it passes through agents and it is hard to find who has modified it.

E. Payment release and obtaining of results by the Client

Having already received the Sample results and the Verifier's report, the Client chooses either to release the Payment voluntarily and receive the full Work results, or to ask to back off and not receive the full Work results – making the choice ends this stage.

Note that the ultimate responsibility of checking the Sample quality is on the Client. The Client may, for example choose to do the check himself, or outsource to specialists. However, the usefulness of a Verifier becomes obvious here. There are reasons why the Client many find it more practical to proceed without centralized checking when W, V agree. Some possible reasons are:

(a) costs of calculations,
(b) there could be cases where having a small percentage of faulty calculation does not create problems, especially compared to variance in reliability of data itself,
(c) transaction costs (having to sign legal contracts, do paperwork, fill orders for multiple small transactions etc.),
(d) the subtleties of complying with international regulations can make people and businesses be unwanted as clients, based on their country of origin, due to censorship reasons etc.

An additional reason for Verifier's existence is having network's own protection against cheating, rather than outsource it to Client's agents who could be corrupted, or regulated into censorship, or having an incentive to act with malice against the network.

F. Payment settlement

1. If the Client has voluntarily released the payment:
   - If W and V were in agreement, then W and V get the pre-agreed payments for their contribution to the Work.
   - If W and V were not in agreement, then the following distribution happens, unless canceled by V within a certain time limit. By this distribution, W gets the pre-agreed payment for his contribution to the Work, and V loses a significant part of the stake (around 75% of v would be an economically logical area that averts cheating, and still leaves enough incentives to V not to go to Arbitration without good reason). This penalty money is mostly burnt; a small part of money might be awarded to the Client for his contribution to network safety and/or for costs to have to re-check the Sample results because of W, V disagreement. If V cancels the distribution within the given time limit, then Arbitration follows III-G.

   Explanation: If the Client has voluntarily released the payment, this means that he desires to receive the results and is ready to pay for them. This also means that the Client agrees with the Worker, not with the Verifier. Hence the process starts by offering to Verifier to surrender most of its stake.
2. If the Client selects that he will not voluntarily release the Payment, i.e. asks to back off (on the grounds that he is not satisfied with the Sample result quality:
   - If W, V were not in agreement, then the following distribution happens, unless canceled by W within a certain time limit. By this distribution, V gets the pre-agreed payment for his contribution to the Work, and W loses a significant part of the stake (around 75% would be an economically logical area that averts cheating, and still leaves enough incentives to W not to go to Arbitration III-G without good reason). This penalty money is mostly burnt; a small part of money might be awarded to the Client for wasting time. If W cancels the distribution within the given time limit, then Arbitration follows III-G.

     Explanation: If the Client does not release the payment voluntarily, this means that he is not desiring to get the results. This also means that the Client agrees with the Verifier (when Verifier claims that the Worker is faulty), not with the Worker. This is why the process starts with the offer to Worker to surrender most of its stake.
   - If W, V were in agreement (a rare case where W and V agree, but not C), then the following distribution happens: all three agents C, W, V receive a small penalty which is burnt. The remainder of Payment is refunded to the Client, while the remainder stakes are returned to W, V. The Client does not receive the results. Arbitration cannot be initiated, because there is a chance that W, V are found innocent and there will be no stakes cuts to cover Arbitration costs.

     There arises a question: if the Verifier was to be selected by the Client, on Client's full responsibility, could we then omit the Client's choice not to release the payment voluntarily when W, V results agree between themselves? This is possible. That way the Client effectively would lose the right to protest, and his interests are represented by Verifier only. However, it might still be beneficial to keep separate the right of agents to protest, so that the faulty Verifier that conspires with the Worker can be deterred.
3. If the Client fails to reply about his choice within the given amount of time:
   - If W and V were in agreement, then W and V get the pre-agreed payments for their contribution to the Work.
   - If W and V were not in agreement, then Arbitration. It may be reasonable (if not too difficult technically) to give them the chance to avoid Arbitration, by agreeing collectively to either of the two distributions: (W paid, V penalized 75%), (W is penalized with 75% of its stake, V is paid).

G. Arbitration (optional)

Arbitration happens only if no agreement about distribution of payment and locked stakes was reached in the previous stage. To become eligible, Arbiters would need to put their own stake as collateral (comparable in size to stakes of W, V), so as to judge responsibly, having skin in the game.

Arbitration is done with as many (random) Arbiters as possible. The limitation to the number of Arbiters is the collateral stake size that pays for Arbiters calculations; the collateral stake of the faulty W or V are distributed among Arbiters. The payment of Arbiters must not be different, whether they exonerate W or V. This is achieved by either having symmetric stakes $w = v$ (right from start or when going to Arbitration), or by fixing Arbiters total payment as being the smallest from v, w. The Client's locked Payment never goes to the Arbiters, but is either refunded to the Client (when V is found true) or awarded to the Worker (when W is found true).

One option is that Arbiters decide the outcome of Arbitration by simple majority, in that case they should be an odd number. Another alternative option is to design incentives for Arbiters, to discourage corruption, making it unprofitable. For example, if 70% of Arbiters agree on any outcome, then they are rewarded and those Arbiters that do not agree are seriously punished by stake cut. If less than 70% Arbiters agree on any outcome (i.e. the agreement rate is between 50% and 70%), then the totality of Arbiters is slightly punished and this money is used to re-do the Work anew or to refund the Client (while W, V get their stakes back untouched).

Also, it is worth noting that when Arbitration starts, most previous choices of C, W, V become irrelevant, e.g. who initiated the Arbitration and on what grounds is irrelevant; the Arbitration is a greenfield approach.

IV. Aligning of economic incentives

Let p be the probability of cheating detection, s the standard payment for work and c the cost to produce the calculation.

Let the standard collateral stake, lost when cheating is detected, be w for Worker node and v for Verifier node, where $v \geq w$.

In order for the network to operate, producing nearly perfect results in equilibrium, cheating must be unprofitable.

A. Assumptions

1. Client wants to pay only if payment is required in order to receive correct calculations
2. Client does not want to pay to receive faulty or missing calculations
3. Worker wants to spend resources to do correct calculations only if its payment is more than the costs incurred
4. Worker may not want to spend resources to do correct calculations, if there is a possibility to get paid for faulty or missing calculations.

B. Prevention of worker cheating: collateral stake size vs probability of detection trade-off

Let's calculate the expected profits from cheating. The payoffs for each world condition are given in the following Table.

Table I: Basic payoff schedule

Decision to cheat	Payoff
True node	$s - c$
Faulty node (undetected)	s
Faulty node (detected)	$-w$

The expected payoff when not cheating is $s-c$, i.e. standard payment for the assignment, minus the incurred cost.

The expected payoff when cheating depends on the probability of detection p%. Hence it is $(1-p)s + p(-w)$, i.e. the expectation of standard payment s without any doing work minus the expectation to lose one's stake collateral deposit w.

Hence the expected payoff for the strategy 1 is as follows:

Table II: Payoff schedule with detection probability p

Decision to cheat	Expected payoff
True node	$s - c$
Faulty node, dissolve if detected	$(1-p)s + p(-w)$

Therefore, cheating is not profitable when $s - c > (1 - p)s + p(-w) \Longrightarrow p(s + w) > c$

Observe that the result depends on all four parameters. It depends positively on the probability of detection p, the standard payment s and the stake w; it depends adversely on the cost of calculations c.

In order to quantify the above results, we need to make reasonable assumptions about the relation between payment s and cost c. This depends on the "markup" (profit margin) $\frac{s-c}{c}$. If $\frac{s-c}{c} = 0$ then this markup is 0%, therefore, $s = c$. If $\frac{s-c}{c} = 1$ then this markup is 100%, therefore $s = 2c$.

The Table below uses the formula $p(s+w) > c$ and shows how high the stake w need to be, relative to calculation cost c, in order to deter cheating.

Table III: Stake dependence on the cheating detection probability p

p	$\frac{s-c}{c}$	w
0.02	0	$49c$
0.02	1	$48c$
0.05	0	$19c$
0.05	1	$18c$
0.1	0	$9c$
0.1	1	$8c$
0.2	0	$4c$
0.2	1	$3c$

Therefore we find that e.g. a combination of Worker stake $w = 9c$ and repetition of calculations (i.e. probability of detection) $p = 10\%$ would be sufficient to make cheating unprofitable, regardless the markup level. Note that the results are not very sensitive to a change in the markup. Instead, they are quite sensitive to a change in probability of detection p%. This is because the biggest impact on incentives comes from losing the collateral stake rather that from losing the payment for work. Hence a simplified, practical rule of thump

$w \geq \frac{c}{p}$ could be used. The tradeoff is between increasing the probability of detection p by costly repetition of calculations, versus having too high a stake that workers cannot afford.

If c is unobservable on the market, we can logically assume that $c \leq s$ (otherwise the worker would operate at loss) and therefore we can use s as proxy for maximum c. Therefore, the rule of thumb becomes $w \geq \frac{s}{p}$.

C. Prevention of Verifier(s) cheating and their optimal number

The probability of detection p is independent of the number of Verifiers that work on a task; it depends only on the percentage of work re-calculated. Any additional Verifier that repeats calculations increases the customer's cost sharply. However, increase in the number of Verifiers could potentially help to prevent collusion between the Worker and the Verifier (two Verifiers would be sufficient to create a Prisoner's Dilemma). Collusion can happen if the Worker could communicate to the Verifier the result that the Worker intends to send to the network. Knowing that result (and assuming that the Verifier believes the Worker's declaration), the Verifier could send the same result without doing the calculations. Since we cannot predict the futuristic technologies that can facilitate collusion (like smart contracts and automated prediction markets), it is easier to approach the problem by calculating the budget that the Worker node has to spend for facilitation of collusion. The Worker's maximum gain from getting paid without procuring any cost is s. However, the gain of Verifier from reporting the Worker's results as faulty is w (the confiscated worker stake), where $w > s$. In addition, as long as $v > w$, it is easy to setup bounty programs where Worker could benefit from luring the Verifier to collusion (by sacrificing w) and then reporting him to the customer's arbitration to gain v. The Prisoner's dilemma already exists between the Worker and the Verifier, without the need for a second Verifier. Hence a rational Verifier will not collude with the Worker, unless of course they both happen to belong to the same actor. But any actor large enough to benefit consistently from such a coincidence is a large enough player to have its incentives aligned with the network's success and would likely not undermine it by cheating.

V. Conclusion

In this paper, we have examined the incentives of network participants to cheat the network by reaping rewards without doing the proper contribution. These observations are not limited to the market for distributed calculations, but may apply to any decentralized economic sector where the free rider problem is present. We have searched for resolutions of this problem by means of the network itself, without any centralized intervention, and we recommend to introduce the skin in the game factor (node stakes). In addition, we recommend two-staged verification and arbitration process that may have variations, but generally is centered on i) detection of the discrepancy in outcomes, ii) attempt to resolve the dilemma cheaply and efficiently, iii) if impossible, then call for crowdsource to protect the network from attack.

Specifically in our model, we find that if detection of cheating is imperfect, it could be profitable to report fake calculations results without calculating, hoping to remain undetected. We also find that, once cheating is detected, arbitration works quite well and does not present a weak point. The most effective attacking strategy is to cheat, but surrender (and not raise) the collateral stake if caught. Therefore it is recommended to increase the network's ability to deter this cheating strategy. This can be done by setting carefully the parameters of the network, primarily by keeping the stakes of nodes high enough, to ensure that those who reach for rewards have skin in the game.

References

[1] The specs for libp2p and associated submodules. https://github.com/libp2p/specs.

[2] Webassembly. http://webassembly.org/, 2016.

[3] The #freeai manifesto. https://manifesto.ai/, 2018.

[4] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In R. N. Wright, editor, *Financial Cryptography*, pages 84–102, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[5] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timon, and P. Wuille. Enabling blockchain innovations with pegged sidechains. 2014.

[6] C. Badertscher, J. Garay, U. Maurer, D. Tschudi, and V. Zikas. But why does it work? a rational protocol design treatment of bitcoin. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 34–65, Cham, 2018. Springer International Publishing.

[7] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. Cryptology ePrint Archive, Report 2016/116, 2016. https://eprint.iacr.org/2016/116.

[8] J. Benet. Ipfs - content addressed, versioned, p2p file system. 2016.

[9] K. Bennett, C. Grothoff, T. Horozov, I. Patrascu, and T. Stef. Gnunet – a truly anonymous networking infrastructure. In *In: Proc. Privacy Enhancing Technologies Workshop (PET*. Citeseer, 2002.

[10] I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.

[11] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf.

[13] D. Fudenberg and J. Tirole. Game theory (3. pr.). 1991.

[14] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 465–482, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[15] N. Houy. The bitcoin mining game. *Ledger*, 1:53–68, 2016.

[16] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):51–81, 1988.

[17] A. Kiayias, I. Konstantinou, A. Russell, B. M. David, and R. Oliynykov. A provably secure proof-of-stake blockchain protocol. *IACR Cryptology ePrint Archive*, 2016:889, 2016.

[18] N. Kshetri. Privacy and security issues in cloud computing: The role of institutions and institutional evolution. *Telecommunications Policy*, 37(4):372 – 386, 2013.

[19] R. Kumaresan and I. Bentov. How to use bitcoin to incentivize correct computations. In *ACM Conference on Computer and Communications Security*, 2014.

[20] J.-J. Laffont and D. Martimort. *The theory of incentives : the principal-agent model / Jean-Jacques Laffont and David Martimort.* Princeton University Press Princeton, N.J. ; Oxford, 2002.

[21] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, 2002.

[22] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[23] M. Orlovsky. Typhon: trustless bitcoin sidechain protocol. https://github.com/dr-orlovsky/typhon-spec, 2019.

[24] J. Teutsch and C. Reitwießner. A scalable verification solution for blockchains. 2017.

[25] B. T. Ward and J. C. Sipior. The internet jurisdiction risk of cloud computing. *Information Systems Management*, 27(4):334–339, 2010.

[26] G. Wood. Ethereum: a secure decentralised generalised transaction ledger. 2014.

[27] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, May 2010.