Hard Problems in Cryptocurrency: Five Years Later

Special thanks to Justin Drake and Jinglan Wang for feedback

In 2014, I made a post and a presentation with a list of hard problems in math, computer science and economics that I thought were important for the cryptocurrency space (as I then called it) to be able to reach maturity. In the last five years, much has changed. But exactly how much progress on what we thought then was important has been achieved? Where have we succeeded, where have we failed, and where have we changed our minds about what is important? In this post, I'll go through the 16 problems from 2014 one by one, and see just where we are today on each one. At the end, I'll include my new picks for hard problems of 2019.

The problems are broken down into three categories: (i) cryptographic, and hence expected to be solvable with purely mathematical techniques if they are to be solvable at all, (ii) consensus theory, largely improvements to proof of work and proof of stake, and (iii) economic, and hence having to do with creating structures involving incentives given to different participants, and often involving the application layer more than the protocol layer. We see significant progress in all categories, though some more than others.

Cryptographic problems

1. Blockchain Scalability

1. Blockchain Scalability

Status: Great theoretical progress, pending more real-world evaluation

.

Scalability is one technical problem that we have had a huge amount of progress on theoretically. Five years ago, almost no one was thinking about sharding; now, sharding designs are commonplace. Aside from ethereum 2.0, we have OmniLedger, LazyLedger, Zilliqa and research papers seemingly coming out every month. In my own view, further progress at this point is incremental. Fundamentally, we already have a number of techniques that allow groups of validators to securely come to consensus on much more data than an individual validator can process, as well as techniques allow clients to indirectly verify the full validity and availability of blocks even under 51% attack conditions.

These are probably the most important technologies:

- Random sampling

, allowing a small randomly selected committee to statistically stand in for the full validator set: https://github.com/ethereum/wiki/wiki/Sharding-FAQ#how-can-we-solve-the-single-shard-takeover-attack-in-an-uncoordinated-majority-model

- Fraud proofs

, allowing individual nodes that learn of an error to broadcast its presence to everyone else: https://bitcoin.stackexchange.com/questions/49647/what-is-a-fraud-proof

- Proofs of custody

, allowing validators to probabilistically prove that they individually downloaded and verified some piece of data: https://ethresear.ch/t/1-bit-aggregation-friendly-custody-bonds/2236

- Data availability proofs

, allowing clients to detect when the bodies of blocks that they have headers for are unavailable: https://arxiv.org/abs/1809.09044. See also the newer coded Merkle trees proposal.

There are also other smaller developments like Cross-shard communication via receipts as well as "constant-factor" enhancements such as BLS signature aggregation.

That said, fully sharded blockchains have still not been seen in live operation (the partially sharded Zilliqa has recently started running). On the theoretical side, there are mainly disputes about details remaining, along with challenges having to do with stability of sharded networking, developer experience and mitigating risks of centralization; fundamental technical possibility no longer seems in doubt. But the challenges that do

remain are challenges that cannot be solved by just thinking about them; only developing the system and seeing ethereum 2.0 or some similar chain running live will suffice.

1. Timestamping

1. Timestamping

Status: Some progress

.

Ethereum has actually survived just fine with a 13-second block time and no particularly advanced timestamping technology; it uses a simple technique where a client does not accept a block whose stated timestamp is earlier than the client's local time. That said, this has not been tested under serious attacks. The recent network-adjusted timestamps proposal tries to improve on the status quo by allowing the client to determine the consensus on the time in the case where the client does not locally know the current time to high accuracy; this has not yet been tested. But in general, timestamping is not currently at the foreground of perceived research challenges; perhaps this will change once more proof of stake chains (including Ethereum 2.0 but also others) come online as real live systems and we see what the issues are.

1. Arbitrary Proof of Computation

1. Arbitrary Proof of Computation

Status: Great theoretical and practical progress

.

This is basically saying, build a SNARK (or STARK, or SHARK, or...). Andwe've done it! SNARKs are now increasingly well understood, and are even already being used in multiple blockchains today (including tornado.cash on Ethereum). And SNARKs are extremely useful, both as a privacy technology (see Zcash and tornado.cash) and as a scalability technology (see ZK Rollup, STARKDEX and STARKing erasure coded data roots).

There are still challenges with efficiency; making arithmetization-friendly hash functions (seehere and here for bounties for breaking proposed candidates) is a big one, and efficiently proving random memory accesses is another. Furthermore, there's the unsolved question of whether the O(n * log(n)) blowup in prover time is a fundamental limitation or if there is some way to make a succinct proof with only linear overhead as in bulletproofs (which unfortunately take linear time to verify). There are also ever-present risks that the existing schemes have bugs. In general, the problems are in the details rather than the fundamentals.

1. Code Obfuscation

1. Code Obfuscation

Status: Slow progress

.

In plain English, the problem is saying that we want to come up with a way to "encrypt" a program so that the encrypted program would still give the same outputs for the same inputs, but the "internals" of the program would be hidden. An example use case for obfuscation is a program containing a private key where the program only allows the private key to sign certain messages.

A solution to code obfuscation would be very useful to blockchain protocols. The use cases are subtle, because one must deal with the possibility that an on-chain obfuscated program will be copied and run in an environment different from the chain itself, but there are many possibilities. One that personally interests me is the ability to remove the centralized operator from collusion-resistance gadgets by replacing the operator with an obfuscated program that contains some proof of work, making it very expensive to run more than once with different inputs as part of an attempt to determine individual participants' actions.

Unfortunately this continues to be a hard problem. There is continuing ongoing work in attacking the problem, one side making constructions (eg. this) that try to reduce the number of assumptions on mathematical objects that we do not know practically exist (eg. general cryptographic multilinear maps) and another side trying to make practical implementations of the desired mathematical objects. However, all of these paths are still quite far from creating something viable and known to be secure. See https://eprint.iacr.org/2019/463.pdf for a more general overview to the problem.

1. Hash-Based Cryptography

1. Hash-Based Cryptography

Status: Some progress

.

There have been two strands of progress on this since 2014.SPHINCS, a "stateless" (meaning, using it multiple times does not require remembering information like a nonce) signature scheme, was released soon after this "hard problems" list was

published, and provides a purely hash-based signature scheme of size around 41 kB. Additionally, STARKs have been developed, and one can create signatures of similar size based on them. The fact that not just signatures, but also general-purpose zero knowledge proofs, are possible with just hashes was definitely something I did not expect five years ago; I am very happy that this is the case. That said, size continues to be an issue, and ongoing progress (eg. see the very recent DEEP FRI) is continuing to reduce the size of proofs, though it looks like further progress will be incremental.

The main not-yet-solved problem with hash-based cryptography is aggregate signatures, similar to whatBLS aggregation makes possible. It's known that we can just make a STARK over many Lamport signatures, but this is inefficient; a more efficient scheme would be welcome. (In case you're wondering if hash-based public key encryption

is possible, the answer is, no, you can't do anything withmore than a quadratic attack cost)

Consensus theory problems

1. ASIC-Resistant Proof of Work

1. ASIC-Resistant Proof of Work

Status: Solved as far as we can

.

About six months after the "hard problems" list was posted, Ethereum settled on its ASIC-resistant proof of work algorithm: Ethash. Ethash is known as a memory-hard algorithm. The theory is that random-access memory in regular computers is well-optimized already and hence difficult to improve on for specialized applications. Ethash aims to achieve ASIC resistance by making memory access the dominant part of running the PoW computation. Ethash was not the first memory-hard algorithm, but it did add one innovation: it uses pseudorandom lookups over a two-level DAG, allowing for two ways of evaluating the function. First, one could compute it quickly if one has the entire (~2 GB) DAG; this is the memory-hard "fast path". Second, one can compute it much more slowly (still fast enough to check a single provided solution quickly) if one only has the top level of the DAG; this is used for block verification.

Ethash has proven remarkably successful at ASIC resistance; after three years and billions of dollars of block rewards, ASICs do exist but are at best 2-5 times more power and cost-efficient than GPUs. ProgPoW has been proposed as an alternative, but there is a growing consensus that ASIC-resistant algorithms will inevitably have a limited lifespan, and that ASIC resistance has downsides because it makes 51% attacks cheaper (eg. see the51% attack on Ethereum Classic).

I believe that PoW algorithms that provide a medium level of ASIC resistance can be created, but such resistance is limited-term and both ASIC and non-ASIC PoW have disadvantages; in the long term the better choice for blockchain consensus is proof of stake.

1. Useful Proof of Work

1. Useful Proof of Work

Status: Probably not feasible, with one exception

.

The challenge with useful proof of work is that a proof of work algorithm requires many properties:

- Hard to compute
- Easy to verify
- Does not depend on large amounts of external data
- Can be efficiently computed in small "bite-sized" chunks

Unfortunately, there are not many computations that are useful that preserve all of these properties, and most computations that do

have all of those properties and are "useful" are only "useful" for far too short a time to build a cryptocurrency around them.

However, there is one possible exception: zero-knowledge-proof generation. Zero knowledge proofs of aspects of blockchain validity (eg. data availability roots for a simple example) are difficult to compute, and easy to verify. Furthermore, they are durably difficult to compute; if proofs of "highly structured" computation become too easy, one can simply switch to verifying a blockchain's entire state transition, which becomes extremely expensive due to the need to model the virtual machine and random memory accesses.

Zero-knowledge proofs of blockchain validity provide great value to users of the blockchain, as they can substitute the need to verify the chain directly; Coda is doing this already, albeit with a simplified blockchain design that is heavily optimized for

provability. Such proofs can significantly assist in improving the blockchain's safety and scalability. That said, the total amount of computation that realistically needs to be done is still much less than the amount that's currently done by proof of work miners, so this would at best be an add-on for proof of stake blockchains, not a full-on consensus algorithm.

1. Proof of Stake

1. Proof of Stake

Status: Great theoretical progress, pending more real-world evaluation

.

Near the end of 2014, it became clear to the proof of stake community that some form of "weak subjectivity" is unavoidable. To maintain economic security, nodes need to obtain a recent checkpoint extra-protocol when they sync for the first time, and again if they go offline for more than a few months. This was a difficult pill to swallow; many PoW advocates still cling to PoW precisely because in a PoW chain the "head" of the chain can be discovered with the only data coming from a trusted source being the blockchain client software itself. PoS advocates, however, were willing to swallow the pill, seeing the added trust requirements as not being large. From there the path to proof of stake through long-duration security deposits became clear.

Most interesting consensus algorithms today are fundamentally similar to PBFT, but replace the fixed set of validators with a dynamic list that anyone can join by sending tokens into a system-level smart contract with time-locked withdrawals (eg. a withdrawal might in some cases take up to 4 months to complete). In many cases (including ethereum 2.0), these algorithms achieve "economic finality" by penalizing validators that are caught performing actions that violate the protocol in certain ways (see here for a philosophical view on what proof of stake accomplishes).

As of today, we have (among many other algorithms):

- Casper FFG

: https://arxiv.org/abs/1710.09437

- Tendermint

: https://tendermint.com/docs/spec/consensus/consensus.html

- HotStuff

: https://arxiv.org/abs/1803.05069

- Casper CBC

: ../../../2018/12/05/cbc_casper.html

There continues to be ongoing refinement (eg. here and here) . Eth2 phase 0, the chain that will implement FFG, is currently under implementation and enormous progress has been made. Additionally, Tendermint has been running, in the form of the Cosmos chain for several months. Remaining arguments about proof of stake, in my view, have to do with optimizing the economic incentives, and further formalizing the strategy for responding to 51% attacks. Additionally, the Casper CBC spec could still use concrete efficiency improvements.

1. Proof of Storage

1. Proof of Storage

Status: A lot of theoretical progress, though still a lot to go, as well as more real-world evaluation

.

There are a number of blockchains planning to use proof of storage protocols, including Chia and Filecoin. That said, these algorithms have not been tested in the wild. My own main concern is centralization: will these algorithms actually be dominated by smaller users using spare storage capacity, or will they be dominated by large mining farms?

Economics

1. Stable-value cryptoassets

1. Stable-value cryptoassets

Status: Some progress

.

MakerDAO is now live, and has been holding stable for nearly two years. It has survived a 93% drop in the value of its underlying collateral asset (ETH), and there is now more than $100 million in DAI issued. It has become a mainstay of the Ethereum ecosystem, and many Ethereum projects have or are integrating with it. Other synthetic token projects, such as UMA, are rapidly gaining steam as well.

However, while the MakerDAO system has survived tough economic conditions in 2019, the conditions were by no means the toughest that could happen. In the past, Bitcoin has fallen by 75% over the course of two days; the same may happen to ether or any other collateral asset some day. Attacks on the underlying blockchain are an even larger untested risk, especially if compounded by price decreases at the same time. Another major challenge, and arguably the larger one, is that the stability of MakerDAO-like systems is dependent on some underlying oracle scheme. Different attempts at oracle systems do exist (see #16), but the jury is still out on how well they can hold up under large amounts of economic stress. So far, the collateral controlled by MakerDAO has been lower than the value of the MKR token; if this relationship reverses MKR holders may have a collective incentive to try to "loot" the MakerDAO system. There are ways to try to protect against such attacks, but they have not been tested in real life.

1. Decentralized Public Goods Incentivization

1. Decentralized Public Goods Incentivization

Status: Some progress

.

The problem of funding public goods is generally understood to be split into two problems: the funding problem (where to get funding for public goods from) and the preference aggregation problem (how to determine what is a genuine public good, rather than some single individual's pet project, in the first place). This problem focuses specifically on the former, assuming the latter is solved (see the "decentralized contribution metrics" section below for work on that problem).

In general, there haven't been large new breakthroughs here. There's two major categories of solutions. First, we can try to elicit individual contributions, giving people social rewards for doing so. My own proposal for charity through marginal price discrimination is one example of this; another is the anti-malaria donation badges onPeepeth. Second, we can collect funds from applications that have network effects. Within blockchain land there are several options for doing this:

- Issuing coins
- Taking a portion of transaction fees at protocol level (eg. throughEIP 1559)
- Taking a portion of transaction fees from some layer-2 application (eg. Uniswap, or some scaling solution, or even state rent in an execution environment in ethereum 2.0)
- Taking a portion of other kinds of fees (eg. ENS registration)

Outside of blockchain land, this is just the age-old question of how to collect taxes if you're a government, and charge fees if you're a business or other organization.

1. Reputation systems

1. Reputation systems

Status: Slow progress

.

There hasn't really been much work on reputation systems since 2014. Perhaps the best is the use of token curated registries to create curated lists of trustable entities/objects; the Kleros ERC20 TCR (yes, that's a token-curated registry of legitimate ERC20 tokens) is one example, and there is even an alternative interface to Uniswap (http://uniswap.ninja) that uses it as the backend to get the list of tokens and ticker symbols and logos from. Reputation systems of the subjective variety have not really been tried, perhaps because there is just not enough information about the "social graph" of people's connections to each other that has already been published to chain in some form. If such information starts to exist for other reasons, then subjective reputation systems may become more popular.

1. Proof of excellence

1. Proof of excellence

Status: No progress, problem is largely forgotten

.

The main alternative approach to token distribution that has instead become popular isairdrops; typically, tokens are distributed at launch either proportionately to existing holdings of some other token, or based on some other metric (eg. as in the Handshake airdrop). Verifying human creativity directly has not really been attempted, and with recent progress on AI the problem of creating a task that only humans can do but computers can verify may well be too difficult.

15 [sic]. Anti-Sybil systems

Status: Some progress

.

There have been quite a few attempts at solving the unique-human problem. Attempts that come to mind include (incomplete list!):

- HumanityDAO

: https://www.humanitydao.org/

- Pseudonym parties

: https://bford.info/pub/net/sybil.pdf

- POAP

("proof of attendance protocol"):https://www.poap.xyz/

- BrightID

: https://www.brightid.org/

With the growing interest in techniques likequadratic voting and quadratic funding, the need for some kind of human-based anti-sybil system continues to grow. Hopefully, ongoing development of these techniques and new ones can come to meet it.

14 [sic]. Decentralized contribution metrics

Status: Some progress, some change in focus

.

More recent work on determining value of public-good contributions does not try to separate determining tasks and determining quality of completion; the reason is that in practice the two are difficult to separate. Work done by specific teams tends to be non-fungible and subjective enough that the most reasonable approach is to look at relevance of task and quality of performance as a single package, and use the same technique to evaluate both.

Fortunately, there has been great progress on this, particularly with the discovery ofquadratic funding. Quadratic funding is a mechanism where individuals can make donations to projects, and then based on the number of people who donated and how much they donated, a formula is used to calculate how much they would have donated if they were perfectly coordinated with each other (ie. took each other's interests into account and did not fall prey to the tragedy of the commons). The difference between amount would-have-donated and amount actually donated for any given project is given to that project as a subsidy from some central pool (see #11 for where the central pool funding could come from). Note that this mechanism focuses on satisfying the values of some community, not on satisfying some given goal regardless of whether or not anyone cares about it. Because of the complexity of values problem, this approach is likely to be much more robust to unknown unknowns.

Quadratic funding has even been tried in real life with considerable success in therecent gitcoin quadratic funding round. There has also been some incremental progress on improving quadratic funding and similar mechanisms; particularly, pairwise-bounded quadratic funding to mitigate collusion. There has also been work on specification and implementation of bribe-resistant voting technology, preventing users from proving to third parties who they voted for; this prevents many kinds of collusion and bribe attacks.

1. Decentralized success metrics

1. Decentralized success metrics

Status: Some progress

.

This is now generally just called "the oracle problem". The largest known instance of a decentralized oracle running is Augur, which has processed outcomes for millions of dollars of bets. Token curated registries such as the Kleros TCR for tokens are another example. However, these systems still have not seen a real-world test of the forking mechanism (search for "subjectivocracy" here) either due to a highly controversial question or due to an attempted 51% attack. There is also research on the oracle problem happening outside of the blockchain space in the form of the "peer prediction" literature; see here for a very recent advancement in the space.

Another looming challenge is that people want to rely on these systems to guide transfers of quantities of assets larger than the economic value of the system's native token. In these conditions, token holders in theory have the incentive to collude to give wrong answers to steal the funds. In such a case, the system would fork and the original system token would likely become valueless, but the original system token holders would still get away with the returns from whatever asset transfer they misdirected. Stablecoins (see #10) are a particularly egregious case of this. One approach to solving this would be a system that assumes that altruistically honest data providers do exist, and creating a mechanism to identify them, and only allowing them to churn slowly so that if malicious ones start getting voted in the users of systems that rely on the oracle can first complete an orderly exit. In any case, more development of oracle tech is very much an important problem.

New problems

If I were to write the hard problems list again in 2019, some would be a continuation of the above problems, but there would be significant changes in emphasis, as well as significant new problems. Here are a few picks:

- Cryptographic obfuscation

: same as #4 above

- Ongoing work on post-quantum cryptography

: both hash-based as well as based on post-quantum-secure "structured" mathematical objects, eg. elliptic curve isogenies, lattices...

- Anti-collusion infrastructure

: ongoing work and refinement of https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413, including adding privacy against the operator, adding multi-party computation in a maximally practical way, etc.

- Oracles

: same as #16 above, but removing the emphasis on "success metrics" and focusing on the general "get real-world data" problem

- Unique-human identities

(or, more realistically, semi-unique-human identities): same as what was written as #15 above, but with an emphasis on a less "absolute" solution: it should be much harder to get two identities than one, but making it impossible to get multiple identities is both impossible and potentially harmful even if we do succeed

- Homomorphic encryption and multi-party computation

: ongoing improvements are still required for practicality

- Decentralized governance mechanisms

: DAOs are cool, but current DAOs are still very primitive; we can do better

- Fully formalizing responses to PoS 51% attacks

: ongoing work and refinement of https://ethresear.ch/t/responding-to-51-attacks-in-casper-ffg/6363

- More sources of public goods funding

: the ideal is to charge for congestible resources inside of systems that have network effects (eg. transaction fees), but doing so in decentralized systems requires public legitimacy; hence this is a social problem along with the technical one of finding possible sources

- Reputation systems

: same as #12 above

In general, base-layer problems are slowly but surely decreasing, but application-layer problems are only just getting started.