EXPLAINING BLOCKCHAIN—HOW PROOF OF WORK ENABLES TRUSTLESS CONSENSUS.

Aleksandr Bulkin
May 3, 2016

Blockchain technology, of which Bitcoin is an example, can be quite hard to understand. Mainly this is because core concepts tend to get lost among the complexity of non-essential details. This article tries to fill the gap between general audience literature that is entirely uninformative to computer professionals and highly specialized literature that is informative, but often overwhelming. It is written for people with some technology background but without in-depth familiarity with this particular field.

The subject of this article is technology of distributed trustless consensus, for this is the one area in which blockchain systems, like Bitcoin, are indeed a major breakthrough. When it comes to other goals, such as distributed data storage, anonymity, transaction verifiability, data obfuscation, shared ledgers, micropayments, high throughput, digital contracts, and so on, cryptographic blockchain systems are, essentially, incidental. Solutions to these problems are well known outside of the blockchain space and, consequently, I will not focus on them here.

The main innovation that Satoshi Nakamoto introduced in his article is using so-called *proof of work (POW)* to create distributed trustless consensus and solve the double-spend problem. POW is not a new idea, but the way Satoshi combined this and other existing concepts—cryptographic signatures, merkle chains, and P2P networks—into a viable distributed consensus system, of which cryptocurrency is the first and basic application, was quite innovative.

Proof of work is a requirement that expensive computations, also called *mining* for reasons which later will become clear, be performed in order to facilitate transactions on the blockchain. To understand the link between computational difficulty and trustless consensus within a network implementing a distributed cryptocurrency system is a serious mental feat. With this writing I hope to help those who are attempting it.

Because distributed trustless consensus is the primary innovation of blockchain technology, we start by understanding what it is. As the term suggests, there are three parts to the puzzle: (1) consensus, (2) distributed, and (3) trustless. Having explained the three terms, I will synthesize them by illustrating the problem that arises when you put all three together. In the process we will see how a distributed ledger based on POW solves this problem.

WHAT IS CONSENSUS?

Consider what happens when people talk to each other. If I say “please pass the salt”, I know that what I mean to convey is the desire to receive the salt shaker. When you hear those words you also understand that this is their meaning. However there is more to this. Implicitly, *you understand that I understand* that this is the meaning and, furthermore *I understand that you understand that I understand* this. Imagine that I am in a country where English is not commonly spoken. I will most likely not say these words, not because they suddenly stop meaning what I know they mean, but because I no longer have assurance of our mutual consensus around the meaning—I no longer know that you know what I mean.

The state of consensus around meaning of language, therefore, is defined as a state where we both say X to mean Y, we both know that we say X to mean Y, we both know that we both know this and so on, ad infinitum. In other words, it requires what people call *common knowledge*.

With money the picture is very similar, but consensus must include not just the transacting parties, but the rest of the social group where a particular currency is used. For me to receive money in payment for services or valuable goods, not only must I recognize what's presented to me as money, but I must also know with sufficient assurance that everyone else will also recognize it as such.

The requirement that money supply be scarce is crucially important. If people accepted frogs in exchange for useful things—we would be collecting frogs. If people took slips of paper with a hand-drawn image of a giraffe we would all be drawing giraffes. The reason why choosing a scarce resource as currency is important is that overall this provides a motivating factor in the larger social system to which we have all become accustomed. A non-scarce money will not motivate people to render services or produce goods. So hand-drawn giraffes as money won't work. This understanding for us is intuitive and places the restriction on what can be used as money.

Consequently, the consensus around money (denoted Cm) can be simplistically described as follows.

I will accept some token or process in payment for valuable goods or services if:

1. … it comes from a scarce supply using one of the accepted means of value exchange and creation.
2. … I expect that everyone else will accept this token as money of comparable value.

I, additionally, believe that

3. … everyone else adheres to (1), (2) and (3)

This is simplistic and is by far not a rigorous account, but for the goal of explaining digital finance it is sufficient. Notice that (3) implies an existing common knowledge of what money means, while (2) provides a requirement for there to be common knowledge around this particular token in fact *being money* according to the consensus laid out in (3). Logicians usually have a field day with these kinds of self-referential statements, but we shall not get distracted by marveling as they do. We shall just plow on.

We must notice an important implication to which we will return later. It doesn't really matter what money is, what specific token is used, as long as Cm is obtained for it. Cm doesn't tell us what valid money is, but it does tell us some things about what it isn't: hand drawn pictures of giraffes it isn't. Let's remember this for now and move on to the next topic.

DISTRIBUTED:

Digital money can solve Cm(1)—the requirement that it be in limited supply—by establishing a central entity that controls the supply. This is straightforward: a database on some server holds the information about who holds what amount of money and when transactions are performed it ensures that a person doesn't overdraw his or her account. This is how digital banking works today.

Leaving trust issues aside until the next section let's consider the difficulty in making this process distributed. The distributed system must ensure consistency, that is when a transfer is made using the information supplied by one of the nodes it must be correctly reflected in all other nodes. This problem is hard but we know how to solve it. We either make things slow, by waiting for the information to propagate across the network (known as

strict consistency), or slightly unreliable by confirming the transaction immediately but reserving the right to cancel it if it encounters a conflict elsewhere on the network (eventual consistency).

TRUSTLESS:

The third and final problem is trust. Trust is implicitly present in digital finance in many different ways. There is trust that the entity that holds your account information doesn't go about randomly subtracting value from it, the *trust of safety.* There is trust that the entity who ensures circulation doesn't go about randomly assigning money to itself out of nowhere—*trust of issuance.* Finally, there is trust that the system in fact ensures consistency of information, that is it performs its main function—*trust of correctness.*

What does it mean, then, to say that a system is trustless? Obviously, we have to trust a system which we use to facilitate value exchange. When we talk about trustless systems, we mean that our ability to trust it does not depend on the intentions of any particular party, which could be arbitrarily malicious. This is a curious proposition, for how can we possibly hope to transact through a bad intermediary in a way we can rely on?

Turns out, as the reader is well aware, trustless systems are not new. For example, email can not be trusted to protect your content for unintended eavesdropping. Yet, this presents little problem to those familiar with modern cryptography. An encrypted message can be safely sent through an untrusted channel, making this a trustless system.

Similarly, cryptography answers the needs of *trust of safety*, because we can require transactions to have cryptographic signatures. If everyone in the systems is able to verify cryptographic signatures and refuses to accept payment without one—the trust of safety is achieved, for it makes it impossible for anyone to transact on your funds.

It is the other two kinds of trust that present a serious problem. But before we can look at that closer, we have to synthesize what we know so far.

POW BLOCKCHAIN:

A digital transaction is some event or process whose effect is to transfer money in the sense we laid out in Cm. Putting together our understanding of consensus around money with our idea of trust, we can now list the requirements we have placed on such process so far (we will refer to them as Tx).

In order for our digital monetary system to work, the recipient of a transaction must be able to confirm that:

1. The originator of the transaction is in possession of the funds being transferred.
2. The originator of the transaction has obtained the funds by one of the means commonly recognized as valid.
3. As an outcome of the transaction the recipient will now be recognized by everyone as being in possession of the funds being transferred.
4. As an outcome of the transaction the sender would not be able to present itself as being in possession of the funds any more.

A user—human being or an automated device, say, a vending machine—uses some algorithm that ensures that Tx (1)-(4) are met. These statements expressed in a natural language describe a combined state between the user and the algorithm. For example, an algorithm may include a verification of a cryptographic signature used in a transaction. A combination of the fact that the user knows this check to be implemented, correctly, within the algorithm and the fact that the check succeeds creates a combined state of knowledge about the validity of the signature. Similarly, Tx(1)-(4) can only be true in a sense of being a combination of the user's knowledge of the

details of the algorithm and the context in which it operates, together with the results of the algorithm's execution.

If it is commonly known that the same algorithm (or, more generally, protocol) is used by everyone who engages in financial transactions then it implies assurances regarding what others' view of the transaction is—namely the views mandated by the protocol. Consequently, whenever the description of Tx speaks of assurances about "others'" views, this is on condition that everyone uses the same protocol.

This condition, of course, makes sense. Using the same protocol as everyone else is, naturally, a prerequisite to ensuring Cm(2)—one's ability to spend money later. Consequently, circumventing the system to some malicious end must be done within the constraints of what this common protocol allows.

BITCOIN APPROACH:

So far we have not said anything about exactly what a transaction is, except to posit a common understanding of its effects. In a centralized system, transactions are simply changes to a central database. Because centralized systems involve trust, Tx(1)-(4) are confirmed through a combination of what the central server reports and the common belief, based on trust, that the central server reports accurate and consistent information.

In a trustless system, however, things are quite a bit more complex. For starters, one can not rely on a report by any single entity. Cryptocurrency developers take an even stronger view—that one can not rely on a report by any number of entities. The reason for this is simple: we assume that there is little to no cost to creating any number of colluding malicious entities, which means that faced with conflicting information deciding whose report to believe may be impossible. (I don't find this premise to be unquestionably true, but that's what blockchain is based on, so there)

The assumption that one can't trust anyone's reports leads to the conclusion that one has to be able to confirm things for oneself. This gives rise to the notion that when presented with a transaction, the recipient must ensure by personally reviewing the entire history of transactions the fact that the sender is in fact in possession of the funds and that the way this came to be is valid. This, in turn, leads us down the path of shared ledgers where the main premise is that the entire history of all transactions is completely open and public.

Here is where things start to get interesting. Didn't I just say that you can't believe reports anyone makes? And if so, what about the server from which I am downloading the ledger? Can't it present me with a doctored picture?

The answer has two parts.

First is that it is much harder to produce a totally self-consistent ledger than a report related to a state of a single account. A self-consistent ledger requires, for instance, that transactions be cryptographically signed by the account holders, making a ledger in which you deduct someone else's money simply impossible.

But producing multiple self-consistent ledgers is still not very hard. For example, you can easily create a ledger that only differs in the last transaction in the account from which you hold the cryptographic key.

So the second part of the solution has to answer the question of what does one do when faced with entirely self-consistent but different ledgers? In the crypto parlance this is referred to as the double-spend problem. It is so called because of the possibility that someone malicious can present two different ledger states to two different users, or even worse—present the entire network with a ledger state and then present the entire network with a different state but one that will be ultimately accepted, negating the first one.

PROOF OF WORK:

The solution that was originally proposed in Satoshi's article addresses this problem. To explain it, we first posit an assumption that the Internet is *eventually connected* that is every state of the public ledger will eventually be observed by everyone. In other words, hiding is impossible for a prolonged period of time.

Next, we point out that the choice between conflicting states is not one where one state is *a-priori* good and another is *a-priori* bad, since we posited that both states are totally self-consistent. The difference between a good state and a bad state is that of consensus only, that is if we can *all* agree on which one is good and simply ignore the other one, we will have solved the problem.

There is slightly more to this, though. Namely, by Cm(2) we require that consensus established in the network persist in the future. If our common way of choosing a good state out of several conflicting ones doesn't ensure this, then someone can present the network with a state of a ledger that shows them to be a sender of some transaction, receive the benefits from the recipient and then present the network with another, "better" state, that negates the earlier transaction.

So what we want is a situation where once we see a state of a ledger and accept it as good, that we must all believe that it will be impossible to create a better one later. And this is where proof of work comes in. Proof of work adds artificial computational difficulty to the ledger. The common protocol requires that of any two conflicting states we observe on the network we select the one that was hardest to generate. We also require that there is a common and high level of hardness to recording transactions in the ledger.

Mining is a process of generating proof of work. The way it works is that all miners compete to find a number that, when added to the block of transactions, causes this block to hash to a code with certain rare properties. Based on the cryptographic features of the hash function used in this process, finding such a rare number is hard, but verifying its validity when it is found is easy. For the ledger to be considered self-consistent we require that every block contain such rare number whose hardness we control based on the size of the participating network (a value expressed as a *hash-rate*).

This ensures that once we observe a valid state of the ledger, transactions that have certain age can not be negated, because producing a longer ledger than the one we see requires the malicious entity to have computing power that can compete with the entire existing network. Consequently, these transactions have lasting consensus regarding their validity.

One final note related to mining is to understand how the limited money supply comes into existence in the first place—how the currency is issued. Bitcoin protocol assigns a preset amount of newly issued currency for every block to the miner who first assembled it with the correct proof of work. This, along with transaction fees, incentivizes miners to perform the work that is required to create consensus for the entire network, keeping the network sufficiently large that circumventing it becomes very expensive. This is why mining is so called—because it can be paralleled to the process of "digging" for new bitcoins.

CONCLUSION:

When presented with a new cryptocurrency solution, the reader is now empowered to ask herself, based on the above explanation, how it is that the solution achieves distributed trustless consensus. Similarly, when considering using a system based on POW, the reader is in position to understand that POW is *only* useful when trustless consensus is required, otherwise it is an expensive and needless waste of resources.

There now exist solutions which do not use blockchains, but still strive to achieve distributed trustless consensus, Iota being one example. Furthermore, there are so-called consortium blockchains, which apply some

of the ideas developed in decentralized cryptofinance to facilitate transactions between mainstream financial entities. While it is not clear what exact approach they will use, it is clear from the above explanation that POW would be a feature entirely unnecessary in such systems, since they don't have to function in a trustless fashion.

With all the hype around cryptotechnology I hope that this article could be helpful in navigating the complexity and diversity of this space. If you have questions, comments or suggestions please feel free to join me on Slack at slack.coinfund.io.