Florin - Improving the blockchain

Malcolm J. MacLeod

malcolm@florin.org

Gordons Bay, South Africa

Abstract

"Great engineering is the art of intelligent compromise" - Dan Watts

The release of Bitcoin and the blockchain technology that powers it has ushered in an exciting new era for digital currencies and distributed computing, seemingly bringing into existence what many had thought impossible; a trustless decentralised currency. However like many great inventions in the past, this progress was achieved not by breaking any rules of nature or limitations that people imagined stood in the way, but rather by taking a long hard look at the requirements and then coming up with a clever new compromise.

Most inventions of any significance contain many compromises and Bitcoin is no exception, as with most groundbreaking new systems, it would be naive and unrealistic to expect that the first iteration would get everything 100% right. It stands to reason that there is room for improvement.

It has been over 12 years since Bitcoin burst onto the scene and numerous competitors have since come and gone, some of them bringing some minor improvements to the table, but overall very little meaningful progress has been achieved in core areas. It is my belief that a sober and proper reflection on the current shortcomings, as well as real solutions to some of them are necessary, or this promising new technology may easily falter while still in its infancy. This article attempts to pinpoint what I believe are the shortfalls and compromises of current blockchain technology, and analyse them in search of ways to improve the system, with the goal of implementing these improvements in our virtual currency Florin.

Keywords: Blockchain, Florin, Bitcoin, Distributed consensus, Hashcash, Proof of Stake, Proof of Work, PoW^2

1. Introduction to the blockchain

The blockchain represents, to date, the best (partial) solution to a very complex problem known in computer science as trustless distributed consensus. Perfect trustless distributed consensus would be the ability for multiple computers to agree on and keep record of an order of events/information, in a manner that is permanent (cannot be tampered with or forged after the fact) but without having a central authority in the system that decides on or controls this order, where all peers in the system are essentially equal and none of them have any special control over the system.

The blockchain is not a perfect trustless distributed system, but it is a trustless distributed system. It achieves this compromise by relaxing one of the criteria slightly, namely instead of history being 100% incorruptible/unforgeable it settles instead for a history that would be incredibly difficult to tamper with or forge with the assumption that when applied to a monetary system this requirement is sufficient. I will touch more on this later in the paper, first I want to focus on the great benefits that this allows:

1. No centralized point of failure, there is no single piece of infrastructure that can be taken down that can cause an interruption of service or a loss of history. Traditional alternatives are very susceptible to this, and we have seen numerous cases in history of banks losing peoples transaction history or e.g. of the Visa network going down and being temporarily unusable.
2. No centralized control, nobody can control the network and tell it what to do, everyone must play by the same rules. This eliminated the possibility for corruption and embezzlement that has plagued the banking industry in the past.
3. No oversight required – In most countries today it is not possible to open a new bank or payment service without complying to mountains of legal requirements and oversight from government, and not without good reason. Without such oversight the central authority can easily make off with everyone's money[1]. Due to (1) and (2), a blockchain based service bypasses the need for all this legislation, allowing for services to be rolled out internationally, faster and cheaper.

While blockchains are certainly not limited to payment systems, or currencies, and there has been of late many attempts to use the same concept for numerous other use cases, this paper is written from the perspective of Florin a digital currency and therefore everything that follows is in the specific context of decentralised virtual currencies and specifically Florin, and should therefore be read as such.

2. The problems

Unfortunately[2] current blockchain implementations fall short of the ideal that people would like from them, or the expectations that people have from a currency and what they expect such a currency to offer. Some of these problems

[1] Not an uncommon thing when one looks for instance at pension funds, and even with the oversight this can still be a problem.

[2] As with most new technologies in their first incarnations, or indeed even most established technologies.

have already become visible/obvious to the public on larger currencies like Bitcoin and some of them may only become obvious at a later date[3]. Many of these problems are inherent or a side effect of the distributed nature of a blockchain and therefore may at best be mitigated, while others are only limitations of the initial implementations and could potentially be overcome. The first step of course is to identify these problems. Below is not a comprehensive list of all possible problems, but a list of problems that the Florin team considers to be the most important to look at, at this point in time.

2.1. Double spends

A simplified description of how this works:

1. The attacker creates a transaction $\mathbf{T}_1$ that sends funds to recipient $\mathbf{R_1}$the 'target' of the attack.
2. He creates a second transaction $\mathbf{T}_2$that sends the same funds to a different recipient $\mathbf{R}_2$[4]
3. $\mathbf{T}_1$ is revealed first to the network, the recipient becomes aware of the transaction and acts upon the receipt assuming that the attacker has made payment and everything is in order.
4. However, $\mathbf{T}_2$ is then also revealed to the network by the attacker.
5. The network can only accept $\mathbf{T}_1$ or $\mathbf{T}_2$ as valid but not both, by ensuring that the network accepts $\mathbf{T}_2$[5] instead of $\mathbf{T}_1$ the attacker has effectively stolen from the original recipient.

In an ideal payment system this would not be possible, however this is unfortunately one of the side-effects that the decentralized nature of the blockchain brings. Some more in depth analysis on the subject can be found in Rosenfeld [12]

2.1.1. Zero-conf double spend

The zero-conf double spend is the easiest way to perform a double spend. It is the easiest/cheapest attack to perform in terms of technology, but the hardest in terms of finding a victim. The victim will know relatively soon[6] that they have potentially been ripped off by an attacker, so the attacker would have to exercise some caution in how/where they did such an attack[7] in order to be able to make an escape without being captured.

[3] As the number of users grow and/or the blockchain or transaction numbers on it grow scaling problems may become more obvious to end users

[4] Most likely, but not necessarily, an address under his own control.

[5] The exact mechanisms at play here differ slightly depending on whether we are attacking a 0-conf transaction or one that has already entered the blockchain, more information on the various methods in the subsections that follow.

[6] Straight after the next block is mined, or even as soon as the conflicting transaction reaches them.

[7] Online targets like exchanges are ideal targets if they are foolish enough to accept 0-conf.

Essentially it relies on finding a merchant who accepts transactions before they have even entered a block in the blockchain. It is then possible to double spend simply by either (a) Mining the next block yourself (b) Convincing a miner of the next block to include your second malicious transaction instead of the first transaction – this could be done by paying the miner, by putting a higher fee on the malicious transaction, by luck, or by various other means.

The reality is that zero-conf transactions are completely unsafe for various use cases (The exception being cases where the merchant is capable of physically tracking down any double spenders and prosecuting – for instance). The general reaction to this is to place the blame on the merchants and to claim that further education can solve the issue. While this is valid to an extent, the reality is that many merchants will continue to accept zero-conf even after being educated as their business model simply can't tolerate waiting long periods of time for even a single conf.

The inevitable losses that such businesses make once somebody manages to rip them off are harmful to the ecosystem as a whole and ideally need to be stopped.

2.1.2. Short chain double spend

This is the most feasible attack for an attacker with more funds, in terms of finding victims etc. The attacker makes a payment and waits for it to be included in a block, he then immediately begins mining a private side chain that has his second conflicting transaction in instead. He carries on mining (at a faster rate than the main network) until he is ahead of the main network, has received the goods from the merchant and is confident of his escape, at which point he unleashes his longer chain. If a merchant accepts 1-conf, then an attacker needs to mine 2 blocks in a row to pull this off. For 2-conf, 3 blocks etc. This is why it is recommended that people wait for 7 confirmations – which makes an attack with anything less than 30% hash rate unlikely – and usually would require >50%. However 7 confirmations is a long time period and as discussed above this is a huge problem for merchants. The reality is that the vast majority of merchants are working on 1-conf. The chances of success here are determined by how much hash power the attacker has, how many blocks he needs to make and an element of luck. By obtaining >50% hash rate (>50% attack) he can almost be guaranteed of success, however attacks with lower hash rate can also be done.

This is particularly a large problem for newer coins, especially in their infancy – as they don't have the luxury of being first, if they share a hashing algorithm with a larger coin a >50% attack against them is much more feasible. As most of the 'good' algorithm choice are already taken and because there is now more interest in the sector, picking or creating a new algorithm does not automatically protect against this either. Bitcoin and other early coins in their infancy essentially relied on or benefitted from the fact that nobody actually wanted to attack them at that point in time, which helped to defend them from this weakness. As new virtual currencies now enjoy much more attention than before and face much more competition with slower growth, they do not have the same luxury. For younger virtual currencies like Florin a more viable solution is needed here than to pretend that everything is okay and hope that we grow past the point where it is an issue.

2.1.3. Long chain double spend

These are the most expensive, and easiest to defend against, so we won't deal with them much in this paper. Essentially this is the same as a short chain spend, except over a much longer period of time, a day, a month or even all the way back to the start of the coin. Because of the long periods of time involved these are the most costly to pull off however they would be completely devastating to the coin in question if they were to happen. Fortunately these are relatively easy to defend against and are notably one of the exceptions where the bitcoin team has conceded that some decentralization is good. By having checkpoints (basically snapshots of the blockchain) built into each wallet release certain blocks that are expected to be present are hard-coded. This prevents an attack from before the last checkpoint. Various other defenses are possible, some of them with some downsides of their own (e.g. 3.7).

2.2. >50% attacks

The largest attack vector against a blockchain is the >50% attack, if an attacker can gain >50% or more of the networks hash rate then this gives him various capabilities. A brief description of some of these below, most of them are discussed in more detail in their own section.

- Censorship; The attacker can deny specific transactions access into the blockchain by not mining or acknowledging any blocks that contain the transaction.
- Denial of service; The attacker can mine empty blocks (2.7.2) thereby denying service to the network.
- Double spend; The attacker can out mine the network with relative ease and thus execute double spends (2.1) at will.

Thus it is relied upon that at all times obtaining 50% or more[8] of the hash rate should be difficult or impossible to achieve. For Bitcoin which is the most famous and largest blockchain based currency this is easy to achieve but for other newer coins this can be difficult.

2.3. Selfish mining

When a miner mines a block, it is generally assumed that he will immediately broadcast it to the network as this is how honest nodes operate. However there isn't really any such restriction, he can delay broadcasting it to the network for

[8] Actually even as much as 30% can be a problem after factors like selfish mining are taken into account.

as long as he wishes[9], but he can immediately start mining a second block that refers to the currently found block even though he has not broadcast it yet. This process is known as selfish mining, in the simplest of cases it can be used by a large miner to gain a slight advantage, by delaying broadcast of all blocks found by even a few seconds a miner increases his chances of finding blocks compared to the rest of the network. However this can also be abused in more sinister ways[10] – a miner with approximately 33% hash rate engaging in selfish mining could in theory (with a bit of luck) obtain enough advantage to execute a >50% attack i.e. a >50% attack does not necessarily actually require >50% of the hash rate. Eyal and Sirer [9] Garay et al. [10]

2.4. Side chains

Related to the above, it is possible for a miner to mine multiple blocks in secret without sharing it with the network. As such he can work on an attack in private with no risk of being exposed should the attack fail, only once his side chain is already a successful side chain[11], it is then shared with the network at which point it is too late for anyone to do much about it. This characteristic can be utilised by those wishing to perform double spends.

2.5. Centralisation of mining

Real world experience has shown that over time the mining of blocks concentrates more and more toward a small number of individuals or pools. This weakens several parts of the system making it easier to attack.

Examples include:

1. DDoS against larger mining pools in order to gain a temporary >50% advantage to execute a double spend (or simply to cause problems for the coin)
2. Malicious pools – These can mine empty blocks (discussed in Denial of Service), aid people with double spends or cause various other problems for the network.
3. Jump pools – discussed in more detail below (Erratic block times)

2.6. Erratic block times

2.6.1. Block target

For various reasons[12] block intervals can be very erratic. Though ideally a block is meant to come in once every 2.5 minutes[13] in reality they often come in much quicker or slower than this. It is not unusual for smaller coins to see five blocks come in the space of a few seconds and then the next block to

[9] The only restriction being that somebody else might get a block out first.

[10] A 'side-chain' for the purposes of a >50% attack is really in a way just a specialised form of selfish mining.

[11] One that is larger than the main chain.

[12] Brief details in the subsections below.

[13] For Florin 2.5 minutes, 10 minutes for Bitcoin.

take 20 minutes, or an hour. This is at best a major inconvenience for new users and a huge cause of support queries. Worse it can be a show stopper for some merchants to accept the currency as waiting this long for safe transfer of funds just won't work for them, and in the absolute worst case this becomes an incentive for merchants to turn to accepting 0-conf transactions.[14]

2.6.2. Inaccurate block times

As a result of decentralisation, there is no real way to enforce accurate times on blocks. While it's possible to restrict blocks coming from the future[15] the same cannot be done in reverse, it is not possible to restrict blocks with a timestamp in the past via a hard limit. The only restriction here is that the median of the timestamp of the last 11 blocks should always increment. This allows a lot of room for miners to mess around with the timestamp when mining a block, which might be done for one of several reasons:

- Machine genuinely has the wrong time.
- In hopes of gaining more money by tricking the targeting algorithm into yielding easier blocks.
- As part of a malicious attack on the system. For example a time-warp attack that allows a miner to mine an abnormally large amount of blocks in an abnormally short amount of time by tampering with the timestamps.
- As a result of the miner engaging in selfish mining.

It is possible to tighten up the forward drift allowance quite a bit and Florin already has,[16] but ultimately this only limits the inaccuracy of the timestamps a little bit it doesn't solve the problem.

2.6.3. 'Jump' pools

Newer coins that are starting out, and that do not have a unique hashing algorithm[17] there is the added problem that a lot of miners will constantly switch their miners between coins to further maximise their mining profit. This can and does lead to situations where the hash rate suddenly spikes and/or drops drastically and the next block[18] can take a much longer time to come in as a result.

[14] Which are provably unsafe and ultimately a sad story waiting to happen.

[15] Bitcoin clients reject any blocks that have a timestamp more than 2 hours in the future.

[16] Timestamp is limited to 1 minute forward from current time, and median of the last 3 blocks.

[17] Which brings with its own unique problems that we will discuss later in the paper.

[18] Or blocks if the difficulty algorithm is slow to adjust.

2.7. Transaction capacity limitation

2.7.1. Block size limit

The system has an inherent limit on how many transactions it can process. Each block is only allowed to be a certain size[19] and the difficulty adjusted so they come in at a fixed interval[20] [21] leading to an overall limit on how many transactions can actually fit into the blockchain in a given time period. If the limit is exceeded transactions may take very long before they enter the blockchain which can lead to difficulties for users. There has been a lot of press coverage on this over the last few years for Bitcoin where this has become quite a regular occurrence. There has been quite some fighting/controversy over what to do about it see: BitcoinWiki [3]. While the target interval and block size limit can both be adjusted to some extent there is a hard limit imposed by the infrastructure on which the network operates that cannot be exceeded, attempting to allow larger blocks than this limit can be catastrophic for the network. Decker and Wattenhofer [7]

2.7.2. Denial of service/empty blocks

Another factor, that is strangely mostly overlooked in all the debates over is the fact that there is no minimum block size. That is miners can mine an empty block or a block with a single transaction in it, even if there are thousands or hundreds of thousands of 'uncleared' transactions waiting in the system to enter a block. This is not just an implementation detail, but rather a part of how the system works, setting a minimum number of transactions per block is not something that could be properly enforced, and even if it could miners could just generate transactions of their own to meet the minimum. A few reasons miners might mine empty blocks:

- To attack the system, hold it ransom, or otherwise make a political statement.
- To gain a mining advantage.[22]

Even at the worst points where over 170 000 unconfirmed transactions were pending on the network, some miners were still mining completely empty blocks blockchain.info [4, 5]. As long as miners can do this there is the possibility that service can be denied simply by mining empty blocks. While there were some arguments in the past that the network could just refuse empty blocks (for instance) the reality is that this is problematic as it opens up other possible ways to attack the network. What is certain is that any coin that is to scale larger and succeed in the long term needs to address this design flaw.

[19] For Florin/Bitcoin 1Mb.

[20] On average, it is impossible of course for them actually to come at a fixed interval.

[21] For Florin every 2.5 minutes, for Bitcoin every 10 minutes.

[22] There is a small but not negligible speed/resource advantage to be had for miners who do not add any transactions to their blocks.

2.8. Sybil attack

If an attacker can surround another target node with only nodes he controls then he can prevent the target node from seeing the true blockchain, and feed only a blockchain of his own to the target. This, however, is more something for people who implement blockchain based systems to be aware of, rather than a serious underlying issue with blockchains themselves. More in depth information on the concept can be obtained in the following paper: Douceur [8]

2.9. Stalled blockchain

There exists the possibility for a blockchain to stall, if the difficulty is driven up too high, followed by a sudden absence of miners the existing miners may struggle or even completely fail to find new blocks. The effect of this can range from a major inconvenience in the best case, to a death spiral of the currency in the worst case.

3. What has been tried so far, and how it has failed

Various attempts[23] have been made by various virtual currencies in an attempt to solve some of the above problems. There is no time or space, nor is it necessarily productive to enumerate or talk about all of these, but the ones that we consider more relevant are briefly discussed below. Note that this is intentionally kept brief, that is a lot of details are simplified and/or intentionally left out as they are not deemed relevant to this paper, so the below is not comprehensive and should not be read as such.

3.1. Alternative hashing algorithms

A relatively common idea in the virtual currency world[24] is to use a different or new hashing algorithm.[25] The idea is that by having a unique algorithm that no other virtual currency has used before you are no longer susceptible to 2.6.3 and that because all hash rate for this algorithm is pointed at your coin[26] >50% attacks are harder to execute as there is not a large surplus of hash rate that can easily be rented or bought.

This, of course, does hold true to an extent, as can be seen with Litecoin which initially introduced the Scrypt algorithm, and though there have been multiple other coins released since that also use Scrypt it has managed for now to retain enough dominance on Scrypt hash rate that a >50% attack would[27] be incredibly expensive to pull off.

[23] Some better thought out and implemented than others, which might be said to be more based on wishful thinking than any sound reasoning.

[24] Perhaps because of how easy it is to do, and how easy it is to market at people with little understanding as if it is a big change.

[25] As the pool of suitable hashing algorithms in the computing world is rather small, this often means inventing a new algorithm, a process that is difficult to do correctly.

[26] Or at least this is what people like to think.

[27] As with bitcoin itself.

However, the exception does not prove the rule, Litecoin was one of the earliest virtual currencies to fork from Bitcoin and managed to gain a large market share/value and hash rate before virtual currencies became as well known as they are now, and before they were as well researched as they are now. When Litecoin was in its infancy there were fewer people with the knowledge required to execute a >50% attack and less financial motivation to do so as there was a lot less capital being thrown around. This 'early mover' advantage, therefore, means that the same can't necessarily work for other virtual currencies, and real world experience has shown that indeed for most it doesn't.

There are several other problems that a new algorithm brings which need to be considered:

- The pool of existing hashing algorithms that have been verified to be randomly distributed and secure is small and almost all if not all of them have already been used by one or more coins.

- If picking an existing hashing algorithm from existing computer science literature that is not yet used for hashcash (Back [1]) based virtual currencies[28] it is impossible to know for sure how much existing hash power is out there. A virtual currency could have 1000 users mining using CPUs or GPUs with users all under the belief that their funds are secure, meanwhile an attacker who has secretly obtained an ASIC, FGPA or botnet could out mine them all with ease and perform one or multiple attacks. The only way to ensure this does not happen is to develop ASICs especially for the virtual currency as rapidly as possible – something which is a huge expense and impractical for most coins.

- If inventing a brand new algorithm, the new algorithm could have flaws that can be exploited. If the hash is not completely randomly/evenly distributed for example an attacker could manipulate the input blocks to gain an advantage. If a flaw were found an attacker could exploit it to hash at a significantly faster rate than anyone else. To make a new hashing algorithm that is proven to be reasonably secure and randomly distributed in a proper way is a huge undertaking one that usually involved multiple experts over a large period of time[29]; it is something out of the reach of most if not all virtual currencies in terms of budget and practicality.

3.2. Alteration of block target

Another very common idea that is seen among virtual currencies is the idea of using a faster block target than that of Bitcoin which is 10 minutes. There are two main arguments that are usually given for this:

1. Assuming the same block size limit more transactions per second can be processed if blocks occur more frequently. Allegedly solving 2.7.

[28] Skeincoin, Qubitcoin etc.

[29] Usually done in a competition format involving peer reviews from multiple experts.

2. The average time a user has to wait to have their transaction confirmed is lower if blocks come in more frequently.

These arguments do hold true to an extent, the faster a user's transaction enters a block the less likely users/merchants are to resort to trusting 0-conf transactions. A 1 Mb block every 2.5 minutes instead of every 10 minutes does imply four times the transaction capacity limit. However, there are limitations to how far this can be pushed; a 1 Mb block takes a certain amount of time to propagate to all nodes on the network based on latency between the nodes, the time it takes to verify the block before it can be passed on, and the time it takes to transfer the 1 Mb of data. This time fluctuates depending on the CPU speed of nodes, the bandwidth between nodes, the number of nodes in the network and various other factors. More in-depth analysis Decker and Wattenhofer [7].

Long block propagation times can have very negative consequences for the network; in the best case it can lead to a higher orphan/fork rate which in turn can lead to a centralisation of mining with likelihood of this increasing as the propagation time rises, in the worst case the propagation time can start to exceed the block target at which point the entire decentralised network can begin to splinter and consensus breaks down. A comfortable margin should be allowed so that on occasions where the network operates slower than normal problems do not occur as a result.

More frequent blocks also mean a lower difficulty target per block and therefore lower security per block, combined with the increase in forking this means the number of confirms users should wait for is much more than the recommended 5[30] that users should usually wait for a Bitcoin transaction.

Another side effect with faster block times is the increased overhead of all the extra header data. While it sounds like a small thing the size of a few million headers quickly starts to add up and bloats the blockchain size, this can have a very negative impact on mobile SPV wallet users that have to fetch all the headers. It can also have an impact on people trying to use full nodes as more hard-drive space is required and a longer chain download, inevitably this means less full nodes are available which in turn has further detrimental effects on the network.

Sadly many virtual currency authors[31] have opted for faster block targets than this, as it makes for a good story to sell and good press if their currency can make claims such as "We can handle as many transactions per second as Visa", what adds to this problem is that the problems are not immediately obvious to users, as long as the network only receives a small amount of transactions per block[32] the propagation times will remain fast so it will appear as if everything works fine. Only later on in the currencies life if/when the transaction volume grows will it be revealed that the claims are essentially bogus.

[30] Or 7 depending on who you ask.

[31] Either out of a lack of deep understanding of the problems involved, or for more nefarious reasons of deception.

[32] Which is the case for most virtual currencies other than Bitcoin.

This is not to say that 10 minutes is the optimal time and that there is no room for changes at all, Florin operates with a 5-minute target which is a good compromise. 5 minutes gives a faster 1-conf and more regular confirms without massively increasing the number of confirms a user should wait for and leaving enough room that propagation times and other issues that come into play with overly short block times should not become an issue.

3.3. Proof of stake

A popular solution that many virtual currencies have switched to is proof of stake. Proof of stake is similar to 3.1 in that it replaces the hash algorithm with a different one, however it goes one step further and instead of relying solely on raw compute power instead involves unspent outputs as part of the hashing process. i.e. in order to mine a miner needs to own a certain amount of coins for the currency in question, with the chances of successfully mining a block altered in some way by the quantity of coins and sometimes other factors.[33]

On the surface this sounds like a fantastic solution, a few of the supposed benefits:

- An attacker would need >50% of all staking coins instead of >50% of computing power to perform an attack.
- An attacker is disincentivised from attacking, as why would you attack a coin in which you hold a significant stake.
- More energy efficient.
- Currency is controlled by people with a vested interest in it's health instead of miners that only want profit.

However upon closer inspection, it becomes apparent that this is not as good as it sounds. Not all of these claimed benefits hold up to scrutiny and PoS suffers from various new problems of its own. Some of the major ones are addressed below.

3.3.1. The 'unlocked wallet' problem

A problem that most PoS implementations face is that in order to stake the private key is needed, thus the majority of PoS wallets allow (or require) the user to leave their wallet in an unlocked state with all private keys sitting unencrypted in memory. This makes users of PoS coins more susceptible to attack and theft of coins via remote exploit[34] as well as trojans/malware etc.

[33] The age of the coins commonly plays a role.

[34] SSL heartbleed style attack for instance.

3.3.2. The ‘nothing at stake’ problem

PoS has a potential flaw that has been described as the ’nothing at stake’ problem, it is commonly misunderstood and therefore many think it is a myth. Sadly this is not true. In essence the problem is as follows:

- Distributed consensus has no real concept of ’the present’ only a chain that constantly moves forwards with each new block representing a step forward in time, it relies on the concept of work to move the chain forward in a manner that emulates time, unlike the real world in the virtual blockchain world it is possible to go back into the ’past’ and rewrite history, by creating a new different chain that consists of more work than the original.
- Ignoring various possible flaws/attacks[35] this works because miners use up real world[36] resources in order to build the chain, to build a new attack chain would also require real world resources and if the attack were to succeed the miners who built the first chain would lose the value they earned in exchange for the resources they expended.
- PoS, on the other hand, makes use of virtual resources to secure the chain, building a second chain[37] can be done with the exact same resources that built the first chain, if the second chain succeeds the miners who mined the first chain only stand to lose the profit they made but their resources are still there, nothing has been expended. And this is where the phrase “nothing at stake” comes from.
- Due to this there is very little incentive to stop even honest miners from mining on multiple chains, and for attackers the incentive is of course even greater.

3.3.3. Bribing

Due to 3.3.2 it becomes possible[38] for an attacker to bribe otherwise ’honest’ miners to participate in their attacks, by paying the miners a slightly higher fee than they would earn otherwise.

3.3.4. Stake grinding

There are many different ways to implement PoS[39], however one thing that they have in common is that there needs to be a selection process or competition process via which the person or people ’mining’ each block is selected, as this process needs to take place in a deterministic and distributed way it must draw on the blockchain history in some way to determine this. The two common implementation methods are:

[35] Which we discuss elsewhere e.g. >50% attack.

[36] Where time travel is not yet possible.

[37] Or even a multitude of chains.

[38] At least in theory, though there may be some hurdles in practice.

[39] Many of them frustratingly complex making them difficult to properly analyse, in what amounts to ’security by obscurity’

1. Select eligible miners via a deterministic ’lottery’ like algorithm, where the blockchain acts as ’random’ entropy and a winner (or winners) is selected using an algorithm based on this entropy.
2. Let eligible miners compete to mine a hash for the block, in a regular PoW like manner, with their stake giving them a ’discount’ on the difficulty of the hash that they need to find.

This process becomes the obvious point for an attacker to try and find an exploit or advantage. Stake grinding is one such flaw, which works as follows, an attacker uses processing power to repeatedly alter/generate a vast amount of ’alternate’ histories going back one or more blocks, until he finds one for which his stake will win more often, an attacker can generate multiple alternate chains in this manner for ’free’ limited only by his processing power. At worst if one party does this it allows that party the potential to gain a huge advantage and thereby attack the chain if he desires, at best all miners do this and the PoS has now essentially degenerated into a somewhat difficult to use and erratic PoW that is at best ’as good’ as PoW but realistically worse as it is not designed to operate in this manner.

3.3.5. Quantity of staking coins

An attacker only needs >50% of currently staking coins, not >50% of all coins in the network, it is impossible to tell how many coins are actually available for staking and thus impossible to tell how hard it is for someone to get >50%. Due to 3.3.1 many users don’t stake or stake erratically reducing the overall security of the coin as a whole.

3.3.6. Use of old private keys

Related to 3.3.2 is the problem[40], that the private keys of wallets now hold value even after the wallet is emptied. To use an example, assume I have the private key for address **x**, I put 1000000 coins into this address and leave them there for two weeks, I then send them to an exchange where I swap them for Euros. The assumption at this point is that the account is empty, therefore I can no longer stake using it as the coins no longer belong to me but the recipient of the transaction.

Unfortunately this is only half true, while in the present the account is empty in the past it is not; therefore an attacker can, after selling his coins for Euros, rewind the chain to a point where he still owned the coins and proceed to try stake a new longer chain, one in which the sale of the coins never takes place... An attacker might use 3.3.3 or 3.3.4 to further aid his chance of success in this case.

The worst part, however, is that it need not even be the attackers own coins as in the above example, people tend to be careless with the security of

[40] Once again revolving around time which is an important and difficult issue in distributed computing.

things that they are no longer using, and also are unlikely to understand or care that their empty accounts may still have a value to an attacker; as such they are unlikely to maintain proper security or erase old wallet copies that held money in the past but are now empty. If an attacker can gain access to such a wallet[41] they can use this to perform an attack while expending almost no upfront resources of their own.

3.3.7. Stake build up attack

Most PoS algorithms implement a concept called 'coin age' whereby the longer an output has gone unspent the larger its staking weight becomes, the reason that this is usually done is:

1. As a claim that this solves 3.3.2 because 'coin age' is now the 'something at stake' - these claims are however dubious.
2. As a claim that this makes mining more fair in that people with fewer coins have a larger chance of eventually staking, thereby allegedly avoiding a situation of 'centralisation' whereby the 'rich' generate more income by staking and eventually come to dominate all coins as a result.
3. So that users can only log in occasionally to stake instead of trying to work constantly, working around the problem described here: 3.3.1

Unfortunately while 'coin age' sounds like a nice idea, and some of the stated benefits are nice, it introduces an unexpected flaw into the system. It is worth remembering that in order to perform a >50% attack an attacker does not need to out mine the system on a constant basis, but only for a period of time long enough to carry out the attack, if an attacker who ordinarily would only have 10% of the hash can temporarily somehow gain a larger weight he can perform an attack regardless. Coin age unfortunately allows exactly this, by carefully creating several addresses and then leaving the coins in them to build up weight an attacker can slowly 'build up' his attack capacity and then wait for the right moment to attack.

A second possible problem with coin age[42] is that users are deterred from using their money as if they do they lose their coin age and thus cannot stake, it is often argued that this leads to a situation whereby all users of a PoS coin 'hoard' their coins leading to poor liquidity, poor distribution and ultimately an undesirable currency.

3.3.8. Problems with SPV wallets

Another relatively large problem with PoS is that it is not possible to verify the validity of a block without the full chain history. The chain history is required to see if the stakers signature is actually valid and/or eligible to be the

[41] In the worst case the old wallet of a large exchange for instance.

[42] Not specifically a technical problem but an economic one, however in the case of blockchains the two overlap.

winning one. This has the side effect that SPV wallets, the wallet implementation used by most lightweight mobile wallets can not be used in conjunction with PoS. In order to have functioning mobile wallets for a PoS based coin it is necessary to make use of 3.7 and/or other potentially not desirable trade-offs.

3.4. Combined PoS/PoW

Some coins like Peercoin – combine PoS and PoW – the theory being that this makes the coin twice as secure. However, in reality, this doesn't hold true, the problem is that the PoW and PoS miners are competing with each other to generate blocks.[43] This, in turn, means more orphans and fewer profits for miners, which means reduced hash rate. This at best means the gains are much less than expected and at worst means that it actually makes the security worse. If multiple PoS blocks in a row is a common sight then only PoS is required to perform an attack, and vice-versa if multiple PoW blocks in a row is a common sight then only PoW is required to perform an attack. In short instead of being as strong as both; it is instead only as strong as the weakest of the two, thus opening the coin up to more attacks[44] and not less.

3.5. Multi-algorithm

Another concept implemented by some coins is to use multiple hashing algorithms instead of just one, miners of the different algorithms compete to mine blocks, with the difficulty for each algorithm adjusted independently when blocks are found in an attempt to balance things in such a way that each algorithm finds blocks.

Though there are no real papers that provide a thorough analysis of the supposed benefits of this the claimed benefits from proponents tend to be as follows:

- Improves decentralisation.
- Reduces the impact of 'jump' pools on block times.
- Improve blockchain security as more overall hashing power is available, five algorithms are five times the hashing power.
- Various other claims, many of them rather outlandish.

While this sounds good on paper, when examined closer it seems these claims don't really hold up, a non-exhaustive list of problems:

[43] Taking a quick look at a block explorer for Peercoin shows cases where 7 or more PoS blocks are mined before 1 PoW one is

[44] Various new PoS attacks like grinding become possible in combination with a normal PoW attack.

- Overall hashing doesn't actually increase.
  The reason for this is as follows, the incentive for miners to mine a coin is financial gain, usually in an external currency e.g. selling the mined coin immediately on the market for USD. When there is 1 algorithm to consider the amount of hash power will reach an equilibrium based on the price **x** people are willing to pay whereby mining is profitable for the miners, otherwise they would stop mining. When you introduce more algorithms; for example, introducing 4 more algorithms to make 5 in total, the price people are willing to pay is not going to magically increase. The result, therefore, is that for each algorithm the price people are willing to pay will become **x/5**, miners of the original single algorithm will be earning 1/5th of what they earned before meaning that in all likelihood 1/5th of the hash rate will remain and the rest will stop mining, likewise for four new algorithms each will attract 1/5th of the hash rate they would if the coin used only that algorithm, leading to a situation where the equivalent hash rate is the same as if just one algorithm were used.[45]

- Difficulties 'balancing' the algorithms.
  People often make the mistake of assuming that the quantity of hash mining on a blockchain is the measure of what secures it; this is an understandable mistake as it is half true more hash is, of course, more secure after all. However the truth is a bit more complicated, the true measure of a blockchains security is more along the lines of 'network hash rate for hash algorithm / total worldwide available hash rate for algorithm' i.e. if our network is mined at 50 000 hashes a second it makes a big difference whether there only exists in the world the capability to mine 60 000 hashes a second or whether there exists the capability to mine 60 000 000 hashes a second, the latter not being very secure at all.
  For a 'normal' single algorithm coin like Bitcoin, this distinction is not that important. It only matters that the network has as much hash as possible, the more hash the more secure so the network always accepts the block that adds the most work to the chain. However, once we introduce multiple algorithms **A1, A2, A3** the distinction becomes important. The network needs to be able to deterministically pick between blocks mined by the three different miners and to do this it is required to decide which is most beneficial for the chain. Is the **A1** block mined at 50 000 hashes a second, the **A2** block mined at 1000 hashes a second or the **A3** block mined at 500 000 hashes a second - as the algorithms all have different performance characteristics it is no longer enough to simply pick the 'largest' hash instead the optimal way to decide is to take $\max(1/\frac{A1}{TotGlobalHashrate(A1)}, \frac{A2}{TotGlobalHashrate(A2)}, \frac{A3}{TotGlobalHashrate(A3)})$[46] sadly this is impossible even for a human to determine for any given point

[45] Except with increased orphaning and other loses explained in the following points that could actually lead to a loss in hash rate.

[46] Where TotGlobalHashrate is the total hash rate that exists for this algorithm in the

in time, never mind deterministically in a reproducible way with the world constantly changing.

And so a compromise is made, a static weighting is assigned in code to each algorithm and $\max(\frac{A1}{StaticWeight(A1)}, \frac{A2}{StaticWeight(A2)}, \frac{A3}{StaticWeight(A3)})$ is used to decide, unfortunately depending on how accurate these weightings are it at best leads to inefficiencies in the system[47] and at worst opens the system up to easier attack.[48]

Users also tend to perceive too many blocks going to one algorithm as an ’imbalance’ and therefore developers tend to try to weight the selection in such a way that all the algorithms get a ’fair’ share of the blocks; unfortunately, this further weakens the security of the system as now the selection criteria for blocks has become a matter of ’perceived fairness’ as opposed to ’which block actually secures the system more against attack’.

- More attack vectors.
  For each algorithm added there are now more attack vectors in the code, five times the hashing algorithms in which to find a weakness, more complex difficulty adjustment code in which to find a weakness etc.

3.6. Master nodes

Dash has introduced an ‘instant payment’ that is supposedly secure, it uses a system of ‘master nodes’ to handle the instant payment. The ’master nodes’ act as an extra control layer on top of the network, communicating among one another. The problem is that in order to function these ’master nodes’ must either act in a centralised way or must solve the same problems that make a blockchain necessary in the first place. It follows reason and is speculated that there are likely a variety of ways in which such a system can be attacked, however, I have yet to find a proper analysis showing the details necessary to determine this. For the sake of brevity, I won’t go into much more detail on this concept except to say that the lack of proof that master nodes can in fact function as claimed is enough to rule them out as a serious solution for now from a Florin perspective.

3.7. Checkpoint server

Many coins have resorted to a checkpoint server, or similar system, to help prevent double spends. This is an extremely effective method, by having a checkpoint server run at a set depth[49] it is possible to prevent >50% attacks on any transaction that has a block depth beyond the checkpoint, making these transactions safe from double spend. There are however some downsides to the system:

world/universe.

[47] Not always picking the best block and thereby not obtaining optimal network security.

[48] If one of the weightings is incorrect then an attacker can gain a huge advantage by focusing his attack mainly on that algorithm.

[49] 3 blocks for instance.

1. If the checkpoint server is compromised, or the person/people running it have ill intentions the checkpoint system itself can be misused to help perform double spends attacks; at essentially no cost to the attacker. Other malicious uses like transaction censoring exist.
2. Some have concerns that a government or authority could therefore force the developers to censor transactions.
3. Because of this and its centralized nature, use of a checkpoint server may render a coin illegal and/or subject it to further regulation in some jurisdictions.
4. The checkpoint server is a centralized point of failure and can be attacked or brought down to disrupt the network.
5. Ultimately you effectively have a centralised system at this point, which defeats a lot of what is meant to be the main allure of blockchain (decentralisation).

While this works well as a solution for 3-conf or more[50], and can be used to ensure that larger targets like exchanges are operated safely from >50% attack it does not provide a solution for regular users who need faster confirmations, or users who don't understand the risks involved.

3.8. Difficulty adjustment algorithms

Many coins – ourselves included, have resorted to increasingly advanced difficulty algorithms to try to keep the block times more stable in the face of jump pools. We have made great progress here with our own difficulty adjustment algorithm, which gives us very stable block times in comparison to other coins.[51] This algorithm incorporates several heuristics to help improve prediction capabilities as well as a fallback mechanism that detects if a block is taking too long and lowers the target difficulty thereby reducing the effects of 2.9.[52]

Despite some success/improvements, ultimately there are limitations to what can be achieved. A difficulty adjustment algorithm attempts to predict the future based on information that is not only imperfect but some of which (block time) is under the control of potentially malicious/bad-faith actors who are financially incentivised to provide inaccurate information in an attempt to game the algorithm, and thus will always be wrong in some instances .

While complete accuracy is obviously never possible due to the statistical nature of the process, the larger problem is the imperfect[53] timestamp information in the blocks which is self-reported by the miners (see 2.6.2) and subject

[50] Assuming you are willing to accept the decentralised aspect of it, and depending on the depth of the checkpoint

[51] Even coins like Litecoin with far more hash power at their disposal struggle to keep their block times as stable as what ours are now, with their blocks per day wildly swinging between 400 and 800. cryptoid.info [6]

[52] This difficulty drop is done in a careful manner that is cognisant of the maximum block drift time allowed, so as to avoid any consensus issues.

[53] At the best of times, utterly wrong at others.

to a fairly large allowance for adjustment either forwards or backward in time, this leaves little room for further improvement and a huge usability problem on regular blockchains. At the time of this writing it is not uncommon to see bitcoin blocks take over an hour instead of the 10 minutes a user would expect, for instance.

4. A proposed solution to many of the problems above

After much consideration of the various issues above, and much trial and error, I have come up with what I consider the most viable/ideal solution to the various issues above, or at least as many of the issues as possible. It is my belief this represents a giant leap forward in the area of decentralised virtual currencies. Most this solution is implemented in our coin Florin, with more improvements underway and to be released in the near future.

4.1. PoW² - an improved successor to PoW

4.1.1. The naive/basic concept

Despite the various problems listed with PoS 3.3 and combined PoS/PoW 3.4, the core idea behind PoS[54] is an interesting/enticing one, and it is no surprise therefore that it has captivated so many people. Though the current ideas/implementations involving it are flawed in various ways, this doesn't mean that the idea itself is not a good one; perhaps it just needs to be applied differently. After much thought on the issues involved, I've come up with an alternative way to re-use this idea in a different way, one that brings more benefits and fewer problems.

PoW² works as follows:

- There are two distinct class of block generators on the network; PoW miners and currency based generators who will from here be referred to as holders.[55]
- Mining of blocks is done by PoW miners in the usual manner that everyone is used to from PoW systems.
- When a miner finds a block it is submitted to the network.
- Nodes validate, accept and relay the block as usual however it does not yet get added to the tip of the chain.
- This block is at this point in what I will call a pre-signed state.[56]
- When receiving a pre-signed block an eligible holder will sign the block using his private key converting it into a signed block.[57]

[54] Involving the stake that users of the system have in the process of securing the system

[55] In order to better emphasize the role, they play in the system.

[56] Like a legal document that has been drawn up but not yet signed

[57] In a constant-time; near instant process, or as near instant as possible.

- The first step of holding involves adding additional data to the block, this includes a holder timestamp[58], any additional transactions[59] (subject to the existing block size limit) and a transaction to pay out the holder fee.[60]

- The holder then attempts to sign the block.[61]

- As soon as a valid signed block is created it is rebroadcast to the network.[62]

- Once peers receive a signed block they add it to the tip of the chain as usual and everything proceeds as normal from here, PoW miners attempt to mine a new block on top of the new tip of the chain, and the cycle repeats.

The above is a relatively simple change to how things work currently, however it has a larger and more important impact than one might think at first read through. Below the key impacts on the system:

- Holders can add transactions to empty or non-full blocks and get fees for it. Unlike PoW miners, holders must actively hold a portion of the currency and as a result they have a vested interest in a healthy network; this acts as an additional incentive to keep holders honest. Holders are therefore highly incentivised to add transactions to blocks they sign whenever possible. This means that the network will almost always be able to operate at full efficiency in terms of transaction capacity/throughput and will not be hampered by empty blocks 2.7 in cases where transactions are waiting to be added to blocks.[63]

- When holding a block a PoW miner can no longer immediately mine a second block that follows this one, as it is necessary to have it signed first. As a result selfish mining 2.3 is no longer feasibly possible without controlling a substantial percentage of both the hash rate as well as coins in circulation.[64]

[58] Allowing for a massive improvement in 2.6.2

[59] Acting as a countermeasure to 2.7.2

[60] Which includes also the transaction fee for any transactions added by the holder.

[61] In a manner that can best be compared as similar to existing PoS systems, except using a holder selection system that has some unique properties including: constant time, minimal latency/delay, deterministic

[62] Note that it isn't really necessary to rebroadcast the PoW part of the block to peers that already have it, only the additional holder/signed portion needs to be broadcast. So there is no additional overhead here, the block doesn't have to be sent between all peers twice.

[63] It is true that empty blocks are still possible, but it would require both the miner and the holder to participate in not adding transactions. For both of them to by chance manage to mine the same block, without a conspiracy among a huge majority of miners and holders, the probability of this is incredibly small.

[64] In order to have a block signed the miner will first need to broadcast to the network at which point everyone will know about it. While it is true that a miner could theoretically be the holder for his own blocks I will detail later why this is not realistically possible.

- This also provides a good resistance toward private side chains 2.4.[65]
- A >50% attack 2.2 by PoW miners becomes incredibly difficult. To achieve a >50% probability of controlling a specific block on a regular PoW coin one 'merely' requires >50% of the hash rate.[66] Using the formula $P(x \cap y) = P(x) \times P(y)$[67] we can see that with PoW² to achieve a >50% chance of controlling a specific block approximately 71% of the hash rate as well as 71% of the coin supply $P(0.71 \cap 0.71) = 0.504$; 90% of hash rate and 56% of coin supply $P(0.90 \cap 0.56) = 0.504$; or 95% of coin supply and 53% of hash rate $P(0.95 \cap 0.53) = 0.5035$ is required. This is a substantial increase in overall network security, even when factoring in a likely drop in PoW hash rate due to the reward for holders. See Appendix B on page 34 for further analysis of this.
- At this point the question of self-interest becomes relevant i.e. whether somebody with between 53% and 71% of the coin supply is going to attack a network in which they themselves hold such a large stake. So attackers are likely disincentivised from attacking the coin to some extent as well.
- The practical implication here is that[68] even a transaction with only 1 confirmation on a PoW² coin can be treated as relatively secure[69], and 2 or 3 confirmations incredibly secure vs a standard PoW coin where at least 6 or 7 confirmations are generally considered desirable. The level of security here is displayed further in Appendix B on page 34 which is produced using modified source code that is taken from the Satoshi whitepaper (Nakamoto [11]) the source code is also displayed in Appendix E on page 37.

Of course, nothing is perfect, and any system especially in its naive implementation comes with at least some down sides, in the case of PoW²:

- More complex code base.
  The extra code to implement PoW² does introduce some extra complexity into the system, and extra complexity always means more room for error. However the added complexity is not great, the functioning of the system is still simple and elegant enough that it can be properly reasoned about and evaluated, so I do not feel like this is a cause for concern. Unlike for instance with some of the PoS solutions out there.
- Extra chance of blockchain stalling.
  Having two types of block generators involved in the system means that

[65] For the same reason as above, and with the same caveat.

[66] Actually as little as 33%

[67] The intersection of disjoint probabilities x and y is equal to the probability of x multiplied by the probability of y.

[68] Assuming a well-distributed coin, enough coin holders willing to participate in the holding process and a few other things that can be guarded against in a proper client implementation - enough connected peers that are not spoofed for instance.

[69] Thereby helping to reduce instances of 2.1.1.

there are now two possible points of failure; the blockchain can now stall either due to a lack of holders or a lack of PoW miners, however as long as we are conscious of this possibility it is not difficult to design the system in such a way as to mitigate this risk.

- Security risks for wallets that are holding
  As with PoS implementations, (3.3.1) holders need a private key in order to sign and this can be a security issue if not adequately addressed. This however is something that can and is addressed through proper system design which we deal with elsewhere in the paper.

- Difficulties with SPV wallets
  PoS implementations have difficulty with SPV mode for mobile wallets (3.3.8). PoW^2 partially inherits this problem; however unlike with PoS this does not prevent SPV implementation. SPV nodes are capable of verifying the PoW part of the block but not the holder-signed portion of it, due to the holding algorithm requiring the full blockchain in order to calculate who the valid holders are. This is adequate for them to function in a secure way with the security slightly downgraded from a fully validating PoW^2 node.[70] It is worth noting that this 'degraded' security is essentially **at worst** on par with what it would have been for a PoW SPV node and if implemented right quite possibly still more secure than the PoW SPV node, so though SPV does not directly gain as much from PoW^2 as a full verifying node it is not harmed by it either and on the contrary still benefits.

Therefore it is necessary to refine the process further to try and address or minimise some of these downsides.

4.1.2. The optimised/full concept

There are several opportunities to further improve on the initial naive concept.

1. Opt-in participation - In usual PoS implementations everyone on the network is able to stake, this has some unfortunate implications. Namely it is impossible to tell how many coins are actively protecting the network vs e.g. coins that are in cold storage, have been lost, are sitting on exchanges, whose owners never open their wallets, or are otherwise not available. Without the ability to measure the total number of staking coins it becomes harder to gauge the expense of a possible attack and thus impossible to really know how secure the network is at any given moment. Worse people who have no intention of staking might be selected to stake[71] leading to erratic block times as a result. An improvement can therefore be had by changing to an opt-in system for holding, whereby only coins

[70] Which is anyway the case with an SPV node.

[71] In the case of a 'follow the Satoshi' style implementation.

in 'special' addresses can be selected as holders, the total number of coins securing the network can then be easily enumerated, suspicious activity monitored for and holders selected only from the eligible pool.

2. Time-based participation - Two arguments often leveled against PoS is that it gives too much power to large coin holders and that unlike PoW (which burns electricity) nothing of value is "at stake" in a PoS system. There is no expense to a staker when he signs a block so what is to stop him from signing competing blocks? One way of dealing this, which ties in with Opt-in participation above is to introduce a time concept to staking accounts. When placing coins in a staking account a user can pick a time period for which the funds will be locked (similar to how in regular banking systems you have fixed period savings accounts), the user will be unable to spend coins from the account until the lock time has expired but will be able to stake, by doing this the user now has something real "at stake" namely the liquidity of his money.

   The time period is factored into the equation determining the stake weight for the account, so users who choose to lock their funds for longer periods of time will stake more frequently, this gives an opportunity for users with fewer coins to out-stake those with more coins, helping to level the playing field to some extent.

   This is also beneficial for the network, users who are willing to lock their coins for long periods of time are more likely to have the long term health of the network in mind and therefore are less likely to attack the network, by allowing such users to stake more frequently the security of the network is therefore improved. For an attacker to succeed in attacking the network he would likely have to lock his coins for a long period of time which is not desirable for an attacker and acts as yet another obstacle for an attacker to contend with.

   Finally with users tying a portion of their wealth up for a fixed term, the decrease in the number of coins that are 100% liquid should bring positive impacts for the currency as a whole. As users are unable to rapidly exchange all of their coins in moments of panic or hysteria, this should lead to a slightly more stable market with a currency that is much less prone to giant spikes and dumps in price and attract users with a more long term mindset. If the reward is well balanced and not excessive these benefits can be had without achieving the undesirable effect of drastically decreasing overall market liquidity and an absence of users who actually use the coin on a day to day basis.

3. Double-key based participation - By changing the staking address system to use two instead of one private/public key pair it is possible to solve the security issue of staking wallets. Each staking address will have two keys associated with it, the first we will call the spending-key and the second the staking-key. To spend funds from the account a signature from both keys is required. To stake only a signature from the staking-key would be required. The wallet can therefore keep the spending-key safely encrypted and leave the staking-key unencrypted allowing the wallet to

stake without any concern that it might be stolen; if the staking-key is stolen an attacker cannot steal any funds, all he can do is stake on behalf of the victim. This improvement should allow more users to participate in staking, and is therefore beneficial to network security.

4. Holding selection algorithm - By carefully adjusting the holder selection algorithm in ways that might put a potential attacker at a disadvantage it is possible to improve further on the security model and make an attack even harder.
5. There exist some other interesting possibilities that PoW² can offer, e.g. the possibility that holders could be allowed to temporarily increase block sizes in certain situations to help address high transaction traffic periods in a safe way and thereby address scaling, however, these sorts of ideas are best discussed later so we will not go into further detail on them now.
6. Signature-based synchronisation - By embedding in each block header a small compressed delta/changeset of the set of eligible it becomes possible for SPV nodes to verify blocks by signature as well. Further even full nodes can, with this additional information verify most (all but the most recent) blocks by signature only. This in turn decouples sync speed from PoW verification speed, this allows the possibility of having a PoW algorithm that is slower to verify while still being able to sync the chain with acceptible performance, removing an obstacle that usually makes GPU/ASIC resistance impossible.

4.1.3. Implementation details for Florin

Florin will be making use of an optimised version of PoW² with the following details.

- Holding will be opt-in, users will create and transfer funds into a holding account' in order to participate in the process.
- These accounts will use special address scripts on the blockchain that serve as an indicator to the network that they are intended to participate in holding, special network rules will apply to these addresses to facilitate the process of holding. More details in Appendix A on page 31.
- Holding addresses will be derived from two key pairs instead of one, the network will allow only normal spend operations with the first key and only special holding related operations with the second. Thereby allowing wallets to always sign blocks when their wallet is open without exposing their funds to any risk of theft or having to enter a password at any point.[72] This also allows for set up of special backup holding devices/software to ensure holding continues if their wallet is offline.
- This allows for third party services that sign on behalf of the user/holder, using their holding key without ever having access to their actual funds.

[72] Spending-key remains encrypted in memory, as all other wallet keys normally would

However the reward portion of the operation can be paid out to any address, allowing such services to take some or all of the reward as a fee, this along with the ease of holding on their own hardware/servers should ensure that no holding service ever obtains an overly large portion of addresses.

- Holding addresses will[73] upon creation have a fixed 'maturity' time period which is set using a future block number at which the address will mature, the maturity period is determined by the user and must be between the maximum and minimum[74] period that the network allows, the network will not accept any spends from the address until such time as the time period has expired however will allow the address to be used for holding during that time.
- The algorithm via which the holder/signer is selected will include both quantity of coins as well as the fixed time period as a selection factor. Thereby putting attackers at a disadvantage as they would have to lock up their money for long periods of time in order to compete with legitimate holders.
- A minimum cool down period of 100 blocks[75] will apply after signing a block before an address becomes eligible to sign again, this ensures that even with a large weighting a user cannot dominate the network in any significant way with a single address. More details in Appendix A on page 31.
- The weighting will be slightly biased towards accounts with more coins and longer time periods such that an address with 50000 coins would have a higher chance of winning any given block than two addresses with 25000 each would (given the same time period). This further penalizes an attacker who in order to succeed with a >50% attack would require multiple addresses thereby reducing the impact of his coins and increasing the expenditure needed in order to succeed.[76]
- The algorithm will not implement any sort of coin weight concept[77] as this is unnecessary to its functioning and would only introduce flaws. Instead, the maturity period will allow those with fewer coins a 'fair' chance to compete by opting to take a longer maturity period to make up for their lack of coins.
- The algorithm via which holders are selected will be a random but deterministic lottery style system, and not a competitive system (like regular

[73] Via custom script commands.

[74] Roughly 2 months to 3 years, see Appendix for more details.

[75] Chosen to match the same 100 block maturity period that is imposed on PoW miners.

[76] To attack a depth of 5 blocks for instance an attacker would require at least 5 addresses, thereby splitting his funds in 5

[77] Thereby avoiding flaws like 3.3.7

PoW or PoS) as the latter is prone to grinding attacks. More details in Appendix A on page 31.

- Holder addresses that have not signed any blocks for a certain period of time[78], will be temporarily removed from the pool of eligible addresses and will be required to perform a 'refresh' transaction (at a fee) in order to become eligible again. This will prevent a build up of non-participating systems that could cause the system to stall.
- The block reward for PoW miners will be 0.025 florin (with halvings every 400000 blocks)
- The block reward for holders will be 0.075 florin (with halvings every 400000 blocks)
- Only transactions included in a PoW portion of a block will be considered as having 1 confirmation, transactions in a holder/signed portion will be considered by wallets as having 0 confirmations until such time as a subsequent PoW block is mined.[79] For some purposes it may be worthwhile to consider transactions in the holder/signed portion as having $\frac{1}{2}$ a confirmation, however, we will not implement this concept yet it is something for consideration at a later date.
- SPV wallets will also only recognise transactions as having 1 confirmation once they have been verified by a PoW portion of a block, note that this is anyway the case with normal wallets but for SPV this is for different reasons as an SPV wallet cannot confirm the holder/signed portion as they don't keep a full blockchain. Due to the fact that potential holders are only drawn from the last 10000 blocks and there is a fixed upper bound on memory/complexity in the process, it will be possible for SPV peers to verify the holder/signed portion as well in future; though more development is still required in that area and will not be pursued for the initial launch. An additional concept of 'trusted peers' will also be introduced into our SPV wallets in future which will aid with both this as well as general protection against Sybil attacks (2.8).
- The difficulty adjustment algorithm will make use of the timestamp from the holder/signed portion instead of the PoW portion to enable better accuracy and remove from PoW miners the ability to tamper with the PoW difficulty.[80]

A more detailed explanation/analysis of the holder selection algorithm can be found in Appendix A on page 31.

[78] Proportionate to their weighting.

[79] More information why in 4.1.5.

[80] The timestamp from both will be used for block acceptance but the holder timestamp only for difficulty adjustment.

4.1.4. An analysis of PoW^2 against the known flaws/attacks faced by existing algorithms

- >50% attacks (2.2) - Network resistance against >50% attacks greatly increased, to the point that 1-conf transactions are secure enough for most purposes.
- Selfish mining (2.3) - Possibility of selfish mining substantially reduced, essentially not possible.
- DoS via mining of empty blocks (2.7.2) - Difficulty of achieving this greatly increased, essentially not possible.
- Erratic/Inaccurate block times (2.6) - Accuracy of block times greatly increased as time is now controlled by the PoS miners and not the PoW miners, which in turn allows for better functioning of difficulty adjustment algorithm.
- PoS insecure private keys (3.3.1) - Private keys secured at all times.
- Nothing at stake issue (3.3.2) - Substantial PoW hash power involved, so PoW hash is at stake.
- PoS Stake buildup (3.3.7) - No coin age is involved in the process so the system is immune to this.
- PoS stake grinding (3.3.4) - Grinding can only be achieved via mining new PoW blocks, as a substantial PoW hash rate is involved grinding becomes infeasible.
- PoS old private keys (3.3.6) - Old private keys effectively hold no attack value, as the amount of PoW hash power required to perform an attack with old keys would be substantial.

4.1.5. Consideration of new flaws/attacks

There exist two new possible 'weak points' in the system:

1. As the holding algorithm selects only one winner there exists the likelihood that at times the winner will not be available to perform his signing duty. The obvious way to address this is to introduce multiple winners into the system as redundancy, this would be more similar to what other previous systems have tried, and unfortunately vastly weakens the system opening it up to grinding attacks and other vulnerabilities.
   Instead, we rely on the PoW miners. When PoW miners find a block they do not stop mining but continue attempting to create competing blocks until they receive news of a signed block, as each new block mined will randomly select a new single holder this provides the variation required to overcome stalling without introducing the vulnerabilities that come with multiple holders. Ordinarily, there would be concerns about how long this process might take. If for example 3 holders in a row were not present and

each block takes 5 minutes then we have a 15-minute wait, if 10 holders in a row are missing 50 minutes etc. It would seem that there is potential here both for accidental stalls as well as perhaps a deliberate DoS attack. However, our difficulty adjustment algorithm; Delta, is not ordinary and already has a special mechanism built into it to safely lower the difficulty in cases where block times are overly long. This assures us that as the wait becomes longer the quantity of PoW blocks mined will become more and more frequent, thereby minimising the delay in such situations.[81] This reduces the stalling to at worst a minor inconvenience instead of a major attack vector.

The opt-in nature of the system, the various mechanisms that make having a large percentage chance of being selected difficult/expensive, as well as the fact that non-participating holders are regularly 'pruned' from the system and need to pay a fee to re-enter the system. All work together to ensure a pool of the 'fittest' holders and would make any long-term sustained attempt at achieving a DoS attack in this way impractical and costly.

2. As the selected holder does not need to perform any expensive work to perform the signing, the possibility exists for a holder to attempt a DoS attack on the network by signing multiple blocks all with varying transactions/data and sending them out to different peers. It is worth noting that this is not specifically unique to PoW^2 but could also be conducted on a normal PoS coin simply by using more powerful hardware like an ASIC miner. It is the case here that an existing flaw has just been made a bit more obvious. Thankfully, it is not overly difficult to deal with this situation simply by putting some network rules in place.

    - Clients should not request signed blocks when the headers contain the same base PoW block as those of a header/block they have already received.
    - Clients should not forward multiple headers containing the same base PoW block.
    - Clients should assign a misbehaviour score for each subsequent header containing the same base PoW block.
    - Clients should eventually ban peers after multiple such blocks.

    In the case where different nodes end up with a different signed block, the network will quickly come to consensus again when the next PoW block is mined.[82]

[81] This subtle detail turns out to be one of the important puzzle pieces that make this concept possible, and the lack of this feature previously is quite possibly what prevented PoW^2 from appearing sooner.

[82] Occasional brief forking is part of how blockchains work so should not be cause for alarm.

5. Conclusion

This paper has looked at various of the challenges faced by virtual currencies, the strength and weaknesses of the blockchain in its current form as well as the various solutions that have been offered by some as possible ways to counter these weaknesses. I have looked at both the positive aspects of these solutions, where such aspects exist, as well as their shortcomings.

Based on this we have developed Florin using an exciting new concept PoW^2, which I consider the next generation step, building on top of these solutions something that while simple in description manages to not only significantly improve on many of the weaker aspects of a traditional PoW blockchain but also drastically enhance the network security; doing so in a way that feels natural and does not introduce massive complexity or new failure modes.

In addition to the immediate improvements that this brings, the addition of a holder in the process makes easier various other future developments on top of the chain, including various new ways to handle off chain settlement layers as well as side chains and other concepts.

Appendix A. Holder selection algorithm

The most critical part of PoW^2 is the algorithm that determines the selection of the holder for a mined block. The process has the following requirements:

1. It needs to be random.
2. It needs to be deterministic, i.e. should not rely on additional network communication.
3. It should be resistant to grinding attacks, not possible to gain a mechanical advantage.
4. It should not be possible to gain any advantage by having multiple accounts.
5. It should not be possible to predict the winner in advance.
6. It should be as light on resources as possible.
7. It should lead to as few forks in the blockchain as possible.
8. It needs to be fast and efficient and should not exhaust huge amounts of memory, it should introduce as little delay into the system as possible.

I have discarded the possibility of a hashcash (Back [1]) based system as this fails to meet criteria 3, 5 and 7. Aside from hashcash the remaining possibility is some kind of random selection using the blockchain as a seed, existing computer science literature has an algorithm that is perfect for the task, used mostly in genetic algorithms and known as 'Roulette wheel selection' (Bäck [2]) it is a perfect fit for the task. See image on page 33 for a brief understanding of how such a selection works.[83]

The algorithm will thus work as follows:

- A valid holder input is any unspent output constructed using a special holding script, that is included in any of the last 10000 blocks of the chain.[84]
- Holder inputs are subject to the following restrictions.
  1. Must be locked for a minimum of 17280 blocks from creation. (Approximately 2 calendar months)
  2. Must be locked for a maximum of 315360 blocks from creation. (Approximately 3 years)
  3. Must have a minimum of 30 coins and a minimum weight of 10000.[85]
- Holding inputs that have been newly created, or are the result of a previous holding operation within the last 100 blocks are excluded.

[83] Note that all in program arithmetic is done using appropriate sized integer types and appropriate basing so as to keep precision while avoiding floating point or other inprecision that can be a source of indeterminism between different architectures.In this paper we do not deal with this aspect of things to keep the math simpler to follow and understand

[84] Constant time requirement to scan backward in the chain does not prohibit pruning of the UTXO.

[85] Subject to adjustment in future.

- Holding inputs are inserted into the array in a deterministic fashion, first by age and second[86] by the quantity and finally[87] by block order.

- Holding inputs are assigned a weighting based on their quantity and the amount of time (in blocks) that they are unspendable.[88] The weighting is designed in such a way that it is in most cases[89] more financially beneficial for a user to gather more of their coins into a single account than it would be for multiple accounts[90], and needs to accomodate the fact that larger accounts are more heavily penalised by the 100 block wait than smaller ones are in order to do this, there are many possible weighting formulas that could be used for this each with various pros and cons under various situations (large coin participation, small coin participation and so forth. The formula settled on for Florin is: $Weight = (((Quantity)+(Quantity^2/Modifier))\times(1+\frac{Time}{288\times365}))$[91] With Quantity being equal to the number of coins multiplied by 100 and the modifier set at 10000. 288 is the number of expected/targetted blocks per day and 365 the number of days in a general year.

- A second pass through of all values is done, any inputs that are older than $\max((\frac{Weight}{TotalNeworkWeight}\times 2), 200)$ are removed from consideration.[92]

- A third and final pass through all values is done, any holder whose weight exceeds 1% of the overall weighting as taken from the second pass, is reduced to 1% of this weighting.[93]

- The sha256 hash of the PoW block is converted to a 256-bit seed integer. The use of the normal scrypt PoW hash for this is deliberately avoided to prevent any theoretical manipulation that could be attempted by means of varying the difficulty within certain ranges.[94]

- A roulette wheel selection is then done to select the winning holder from the array, with the spin always starting from 0 to allow for more efficient calculation.[95]

[86] In the case of identical age

[87] in the case of identical quantity

[88] This is set by the user on address creation.

[89] Ideally up until at least 0.5% of network weight keeping in mind the 1% maximum weight rule.

[90] This is important to diminish the effect an attacker can have on the network

[91] In actual code implementations, calculation must be rebased to avoid floating point and therefore the implementation is a bit more complex than this, however left as is here for clarity.

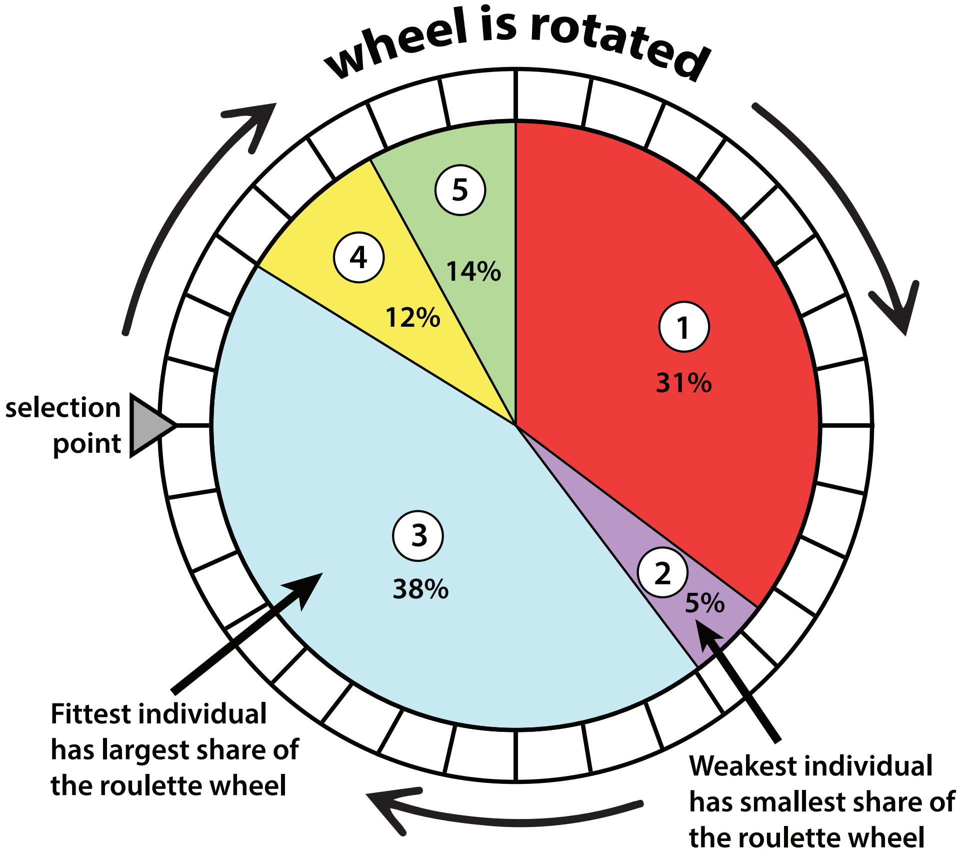
[92] This is to prevent stale inactive holders from stalling the chain repeatedly.

[93] As this change affects the final overall weighting the actual result will be slightly different than 1%, but this is fine no attempt is made to adjust for this.

[94] Even though this is unlikely and the only legitimate case I can think of is when the difficulty gets very high.

[95] Algorithmically we just take the modulus, $Seed\%TotalWeighting$ and then binary search the array for the appropriate place.

This meets all of our requirements, it is completely random and deterministic[96], there is no opportunity for grinding[97], there is a maximum cap on algorithm complexity and resource usage, splitting coins into multiple accounts always give less overall weighting thereby weakening attackers, it is impossible to predict a winner in advance of the block arriving, there is almost no time delay in accepting a holder. As only one holder is selected no forking takes place in this part of the algorithm.


wheel is rotated
selection point
1
31%
2
5%
3
38%
4
12%
5
14%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

[96] So long as we take care to avoid floating point

[97] Except in the PoW generation, but we address that elsewhere in the paper.

Appendix B. Attack probability *[Naive]*[98]

PoW² combines regular PoW along with the holding power of coin holders, in a way that should lead to a minimal drop in PoW hash rate while at the same time substantially enhancing the blockchain security and also bringing some other desirable properties to the table. As it is difficult to grasp how significant the change is, it is best to have a proper visualisation. To illustrate this I've taken code from Nakamoto [11] (see Appendix B), modified it slightly for our purposes[99] and run it to generate a visualisation on page 36.

For this process, I am using the following figures, taken at the time of writing, they are not 100% accurate but for our purposes are more than sufficient:

- Florin network hash rate (SIGMA) 1 gh/s.
- The cost to rent cpu power is approximatels $0.80 for 10Mh/s[100] mining rigs, so 1gh/h = $80[101]
- The cost to purchase cpu power varies wildly, but if we assume second hand hardware $120 for 10Mh/s is possible, or $12000 for 1 gh/s
- Florin price 1 XFL = $70
- Florin availability on the largest exchange, 1000 XFL.
- Florin in circulation 134 000.

The following assumptions are made:

- 50 000 coins participate in holding. [102]
- We calculate based on the attacker using rental hash as this is cheaper than purchasing hash and is possible at the sort of hashrates the network (or a similar equivalent network) would likely have. For larger hashrates an attacker would need to purchase hardware instead which would cost substantially more.
- We will ignore the market effect that buying larger amounts of coins, in order, to attack PoW² would have on the coin price.[103] This would drive up the cost of acquiring further coins for the attack, as well as affect the network hash rate.[104] Therefore the cost estimates for the various PoW² attacks, especially the ones involving larger amounts of coins are likely vast underestimates and would cost drastically more in reality.

[98] Biased toward PoW.

[99] Original core equation unchanged.

[100] Rough estimate, prices and hash of available machines may vary over time and provider

[101] All prices for this Appendix are in USD.

[102] Based on current participation at time of writing, will vary over time

[103] Largest exchange has only 5 000 000 coins available for sale at the moment, so any purchase of more than 3 000 000 coins is likely to have a measurable impact on the market price

[104] The more coins are worth the more network hash rate is attracted.

Attack probability conclusion:

The chart (on the following page) clearly illustrates that even with the comparison done to favour PoW in every possible way, PoW² vastly outperforms PoW in terms of security for any given attack price, and that further even in a scenario where vastly more money is thrown at an attack PoW² continues to offer protection where PoW would not. Even at $1'030'534[105] PoW² remains secure while with only $137700 it is possible to gain complete control over the equivalent PoW network. Even at 60% hash rate and 60% coins there is only a 20% risk at 9 confirms and with a few more confirms a low enough risk to be good enough for most transactions.

For the same attack price point[106] as a >50% attack on the PoW network, the PoW² network yields less than a 1% chance of success to the attacker, leading to what I am terming '**secure 1-conf**' transactions[107] for most purposes and 2 or 3 confirmations being reasonably secure for all but the most sensitive of transactions.

Appendix C. Analysis of attack probability with grinding

The astute will notice that we have of course ignored a possibility above. Instead of opting for e.g. 60% of the PoW hash and 60% of the holding coins an attacker could instead attempt a grinding attack. He could aim to have many multiples of the network hash rate and a smaller portion of the coins. As an example lets say he were to hire 200x the network hashrate, 130 gh/s for $10'400 and 0.3% of the coins $10'500, a total of $20'900.

The probability is as follows:

- $P\left(signingblock\right) = 1 - P(notsigningblock)^{numattempts} = 1 - P(0.997)^{200} = 1 - 0.548 = 0.452$

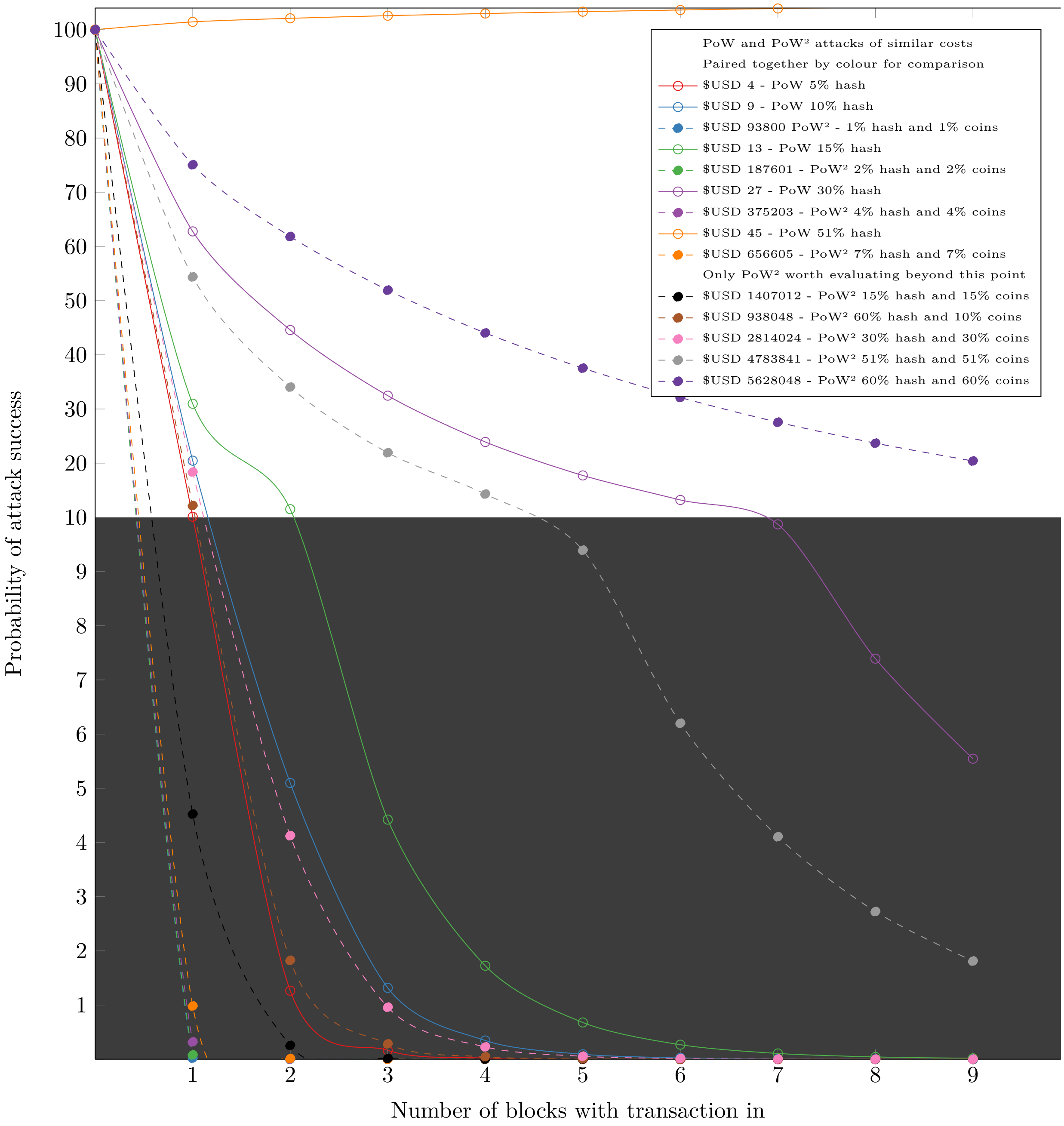
A roughly 45.2% chance of signing a single block, instead of the 0.3% his coins would normally entitle him to, however at a substantial expense compared to a plain >50% attack. While grinding is possible to an extent with PoW², it is resilient to it to the point that it is not likely to be effectual when reasonable network hash rates are involved.

[105] And this is actually an underestimate on the price as this many coins would really cost far more to acquire.

[106] Which in reality would likely cost even more.

[107] Where secure is relative as it has always been, it is common for instance for Bitcoin users to refer to 5, 6 or 7 conf as 'secure' but the security is relative as can be seen in the probability analysis graph below.

Appendix D. Attack probability analysis graph*[Naive]*[108]



[108] Biased in favour of PoW.

Appendix E. Attack probability source code

The below code (borrowed and adapted from Nakamoto [11]) calculates[109] in function AttackerSuccessProbability the chances of success an attacker has when attacking a chain of depth **q** for probability **z**; for PoW we feed in the percentage hash rate as the probability, for PoW² we use $P\left(PoW \cap Witness\right) = P(PoW) \times P(Witness)$. Ignores that a Holder cannot sign multiple blocks in a row, further reducing the probability for each subsequent block as well as various other enhancements discussed in the paper, therefore results are biased in favour of PoW.

```
#include <iostream>
#include <iomanip>
#include <math.h>

double pricePerGh = 90.0;
double networkHashRate = 1.0;
double pricePerCoin = 70;
double networkNumCoins = 50000.0;

double AttackerSuccessProbability(double q, int z)
{
  double p = 1.0 - q;
  double lambda = z * (q / p);
  double sum = 1.0;
  int i, k;
  for (k = 0; k <= z; k++)
  {
    double poisson = exp(-lambda);
    for (i = 1; i <= k; i++)
      poisson *= lambda / i;
    sum -= poisson * (1 - pow(q / p, z - k));
  }
  return sum * 100;
}

int AttackerCost(double percentHashrate, double percentCoins)
{
  int hashCost = pricePerGh * ((networkHashRate*(percentCoins >0.0?0.9:1))*percentHash
  int coinCost = pricePerCoin * (networkNumCoins*percentCoins);
  return hashCost + coinCost;
}

void printPOWAttack(double percentage)
```

[109] In a slightly naive but still useful way.

```
{
  std::cout << ”[PoW] ” << (int)(percentage ∗ 100) << ”% hash\n”;
  std::cout << AttackerCost(percentage, 0.0) << ”\n”;
  for(int i=0;i<10;i++)
  {
      std::cout
     << ”(” << std::fixed
     << std::setprecision(12)
     << i << ” , ”
     << AttackerSuccessProbability(percentage, i) << ”)\n”;
  }
}

void printPOW2Attack(double percentagePOW, double percentagePOS)
{
  std::cout
    << ”[PoW²] ”
    << (int)(percentagePOW ∗ 100)
    << ”% hash ”
    << (int)(percentagePOS ∗ 100)
    << ”% coins\n”;

  std::cout << AttackerCost(percentagePOW, percentagePOS) << ”\n”;

  for(int i=0;i<10;i++)
  {
    std::cout
      << ”(” << std::fixed
      << std::setprecision(12)
      << i << ” , ”
      << AttackerSuccessProbability(percentagePOW ∗ percentagePOS, i) << ”)\n”;
  }
}

int main()
{
  printPOWAttack(0.05);
  printPOWAttack(0.1);   printPOW2Attack(0.01, 0.01);
  printPOWAttack(0.15); printPOW2Attack(0.02, 0.02);
  printPOWAttack(0.3);   printPOW2Attack(0.04, 0.04);
  printPOWAttack(0.51); printPOW2Attack(0.07, 0.07);
  printPOW2Attack(0.15, 0.15); printPOW2Attack(0.6, 0.1);
  printPOW2Attack(0.3, 0.3); printPOW2Attack(0.51, 0.51);
  printPOW2Attack(0.6, 0.6);
  return −1; }
```

Nomenclature

Hashcash Hashcash is a proof-of-work system used to limit email spam and denial-of-service attacks, and more recently has become known for its use in bitcoin (and other cryptocurrencies) as part of the mining algorithm.

Holding In the context of PoW^2 a holder is a person who owns coins and places his coins into a special time locked account and then uses these locked coins to sign PoW blocks as valid.

PoS Proof of Stake.

PoW Proof of Work.

PoW^2 Proof of Work 2 (or Proof of Work squared). The name for our new system that achieves massive security gains by multiplying together the security of Proof of Work and of the holding signatures.

Scrypt In cryptography, scrypt (pronounced ”ess crypt”) is a password-based key derivation function created by Colin Percival, originally for the Tarsnap online backup service.

SPV Simplified Payment Verification, a lighter/faster method used by most mobile wallets. Nakamoto (2009)

References

[1] A. Back. *Hashcash*, May 1997. URL http://www.cypherspace.org/hashcash/.

[2] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996. ISBN 0-19-509971-0.

[3] BitcoinWiki. Block size limit controversy - Bitcoin Wiki. URL https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[4] blockchain.info. Bitcoin Blocks At Height 465740, . URL https://blockchain.info/block-height/465740.

[5] blockchain.info. Bitcoin Blocks At Height 465754, . URL https://blockchain.info/block-height/465754.

[6] cryptoid.info. Litecoin Explorer. URL https://chainz.cryptoid.info/ltc/#!overview.

[7] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013. doi: 10.1109/P2P.2013.6688704.

[8] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44179-4. URL http://dl.acm.org/citation.cfm?id=646334.687813.

[9] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *CoRR*, abs/1311.0243, 2013. URL http://arxiv.org/abs/1311.0243.

[10] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. *The Bitcoin Backbone Protocol: Analysis and Applications*, pages 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-46803-6. doi: 10.1007/978-3-662-46803-6_10. URL http://dx.doi.org/10.1007/978-3-662-46803-6_10.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009. URL http://www.bitcoin.org/bitcoin.pdf.

[12] Meni Rosenfeld. Analysis of hashrate-based double spending. *CoRR*, abs/1402.2009, 2014. URL http://arxiv.org/abs/1402.2009.