Dolla Technical White Paper

Andreas Holmes, Andre Knispel, Maximilian Lupke,
Guidon Malamud, Henning Sauter of Team **Lead One**

and

Team **FP Complete**

Working Draft
Revision 1.1
2018-10-08

Abstract

Dolla is a new cryptocurrency aimed at fast online and offline payments. It is designed to provide

- low latency (~1 second for a transaction to clear) and
- high throughput ($>$ 10000 transactions per second).

In contrast to many other blockchain based cryptocurrencies in which received payments may disappear retroactively with some probability, Dolla is final, meaning that its transactions will never be rolled back and there cannot be any forks in its blockchain.

This is achieved using a permissioned, but decentralised consortium system based on Byzantine consensus with voting. The consensus is a slightly modified version of the DBFT algorithm (see [Cra+18]), which allows n consortium members to agree on a new block in an expected constant number of message broadcasts as long as $t < n/3$ members are Byzantine (faulty, malicious or behaving arbitrarily).

In contrast to many other permissioned cryptocurrencies, Dolla is decentralised and not controlled by some single company or individual. The consensus members decide, per majority vote, which new members to admit and which members to remove from the consortium.

Users of the cryptocurrency need not be consortium members. To append a transaction to the blockchain, a user sends the transaction to the consortium members which verify its legitimacy and inclusion into a block.

While users are anonymous, consortium members are not; they are under public scrutiny. The process of selecting the initial consortium members is carefully designed to ensure a diverse distribution across geography, legislations, political systems and so on. This reduces the chances of members colluding.

Dolla's code and logic is developed using high-assurance methodologies. The reference implementation is written in the Haskell programming language, focusing on correctness and auditability. We strengthen the consensus algorithm presented in the original [Cra+18] paper by the formal verification of its theorems, using machine proofs via the Coq theorem prover. Furthermore, we are currently in the process of proving the Haskell code to be in agreement with the paper using the *hs-to-coq* tool (see [Spe+17]). This provides users and developers with a high degree of confidence about the correctness of both the general approach and the implementation.

Contents

Goals

Current cryptocurrencies suffer from a variety of problems that have so far prevented blockchain technology from displacing established, centralised payment systems such as Visa and PayPal.

In a rough summary, current cryptocurrencies are either slow, not final, centralised, or have low-quality implementations, wherein

- **slow** means that they process many orders of magnitude less transactions than non-blockchain systems[1] and function only because they operate at small scale (for example, Bitcoin handles roughly 3 to 6 transactions per second).
- **not final** means that you can never be sure if money you received will remain yours. Due to the possibility of forks in the blockchain, there is always some probability that received funds will disappear retroactively. This poses a risk of fraud, which can make merchants wary and thus hinder adoption of cryptocurrencies.
- **centralised** means that they are under control of a single entity (usually a corporation) that can change the rules at will and without transparency; consequently also typically under control of the legislation of a single nation that can change the rules to foster their political agenda.
- **low quality implementations** means that the program code that executes their idea is poorly written or maintained, suffering from mistakes that lead to security holes and outages, or make algorithmic analysis and public scrutiny difficult.

Dolla is a new cryptocurrency that takes on the challenge to address these weaknesses.

Moving large-scale electronic payments to blockchain

Most current cryptocurrencies cannot be used by merchants instead of or alongside established payment processors such as Visa and PayPal because they either do not provide the performance or the guarantees necessary for this task. Concretely:

Performance. Visa can process thousands of transactions per second (TPS) and a single transaction completes within seconds. In comparison, Bitcoin delivers around 3 to 6 TPS and it currently takes over half an hour for a Bitcoin transaction to clear. The low throughput means that systems like Bitcoin only work as long as a small number of people use them, and the high latency means that they cannot be used for applications like buying goods in a shop.

Guarantees. After having received a Visa or PayPal transaction, a merchant can be

[1] See https://howmuch.net/articles/crypto-transaction-speeds-compared for a simple comparison of the transactions per second (TPS) througput of cryptocurrencies and conventional payment systems.

sure to truly be in possession of the funds paid[2]. This is not the case for those blockchain systems which allow forks, as forks can undo transactions that occurred in the past[3]. This lack of finality leaves merchants at risk.

We believe that this lack of performance and guarantees makes blockchain technology miss out on one of its biggest applications: Replacing established payment systems with a decentralised, transparent and consequently more trustworthy alternative, and thus becoming mainstream.

For a cryptocurrency to gain real world acceptance as a payment system it must be able to support millions of users at the throughput and latency of Visa, and provide comparable guarantees. To this end, the Dolla blockchain is designed to handle 10,000 TPS at a latency of a second, while its ADBFT consensus algorithm guarantees finality.

High throughput, low latency

As mentioned above, the Dolla blockchain is capable of 10,000 TPS, and during operation the average transaction takes 1 second to clear. For systems with security properties similar to those of Dolla,

- **throughput** is typically limited by the amount of data that needs to be communicated between participants of the consensus, and the available network bandwidth, and
- **latency** is typically limited by the number of message round-trips necessary between the participants of the consensus, and the geographical distance between them.

One advantage of the DBFT consensus algorithm used in Dolla is that it provides strong guarantees on the number of roundtrips necessary. The modifications made in our amended version of the algorithm, ADBFT, are designed to reduce the amount of data sent, thus increasing throughput.

Latency and throughput of a cryptocurrency can be dependent on various factors. In Proof-of-Work systems like Bitcoin, they depend also on a miner mining a new block by solving a cryptographic challenge. Completing this challenge is a probabilistic process; while designed to take *on average* a certain amount of time, this duration is not guaranteed. This makes the actual clearance time of a Bitcoin transaction unpredictable. Dolla is designed such that clearance time depends only on the network latencies between the consensus nodes and not on unpredictable processes during normal operation.

Dolla's design naturally allows for the implementation of nodes as multi-machine clusters

[2] In absence of mechanisms such as credit card chargebacks. However, such mechanisms are explicitly designed into the systems for the purpose of consumer protection, and could be chosen to be left out. In contrast, the non-finality of forking blockchain systems is an inherent problem and cannot be chosen to be left out.

[3] Even if a merchant is not the target of an attack, their funds can disappear as collateral damage of a double spend attack. See the later section on 51% Attacks for details.

in order to maximize the throughput of the consensus mechanism, thus mitigating resource constraints that can limit throughput (e.g. bandwidth or processing power).

Fork-free blockchain with immediate finality

Forks. A common blockchain design is that all participants are supposed to consider the longest chain of blocks they can currently see as the source of truth (or "history"). This chain determines which transactions have occurred. The fact that different participants can observe different chains as the longest (due to network propagation delays) means that the source of truth for any given participant can vary over time. New blocks are typically appended to the longest chain. If two participants observe two different chains, and it happens to be their turn to append a block to the chain at roughly the same time, they make a *fork* in the blockchain, thus creating two different histories containing different transactions. The branches of the fork may grow independently for a while, but eventually one of them will outgrow the other and all participants will switch to the longer one. Switching forks is problematic because it rewrites history: Funds received may suddenly disappear because the transactions containing them do not exist on the other fork. Forks may come into existence by accident as described above. But for some blockchains, especially Proof-of-Work blockchains such as Bitcoin, malicious participants may (through significant expenditure of computational resources) actively create forks to undo previous histories of transactions. There are many recent examples of this being actively exploited. This creates risk and uncertainty about whether funds were really irreversibly received, and reduces trust in the system.

Finality. Finality is a property of blocks. It means that there exists a point in time after which the block is irreversibly committed to the blockchain and cannot be undone. Once this happens, both the block and all transactions within it are said to be *final*. *Immediate finality* describes a relation between blocks: A blockchain has immediate finality if for two blocks A and B, with B being a successor of A in the chain, B cannot be proposed before A has been accepted as final.

Most current blockchains use a design that allows forks and that does not provide (immediate) finality.

Dolla's design makes the existence of forks impossible[4]. A certain quorum of consensus members agrees on each block to be appended to the chain in a lockstep fashion. The source of truth on what is history is always determined by querying a quorum of consensus members. A Dolla block is not considered committed until the quorum has decided so. Once it has, the block and the transactions become irreversible. Dolla provides immediate finality.

[4] *Manual* forks, in which users create another instance of the blockchain using the same code on purpose, are of course always possible.

Full decentralisation

To be able to deliver high throughput and low latency and provide a fork-free blockchain with immediate finality, Dolla is designed as a consortium blockchain (also called "permissioned blockchain").

In a consortium blockchain there is a clear separation between the set of end users that use the cryptocurrency, and the set of consensus participants (the consortium) that coordinate the appending of blocks to the blockchain. One cannot just choose to join the consortium; instead, the potential member has to acquire somebody's permission to join (see below).

In contrast, "public" (also called "unpermissioned") blockchains, allow anybody to participate in appending blocks to the chain. For example, in Proof-of-Work systems like Bitcoin, anybody can choose to become a miner by solving cryptographic challenges to mine blocks. This also makes the system decentralised: There is no central authority to permit or refuse who can participate in maintaining the blockchain. Decentralisation is desirable because it makes systems more robust and trustworthy because one does not have to assume good faith of some central authority. Unfortunately, it is very difficult to achieve high throughput, low latency and immediate finality with public blockchains.

Most consortium blockchains are not truly decentralised. Many are under tight control of some single entity that decide who can be part of the consortium. This effectively puts the consortium members under control and rule of that single entity.

Dolla is decentralised, because it is designed such that no single entity controls the consortium. Anybody can apply to the consortium to become a consortium member. The consortium itself decides by public vote who can join or must leave the consortium.

Evolvability

There are many examples of situations where cryptocurrency communities could not agree on new rules and forked as a result; perhaps most prominent are the fork of Bitcoin Cash from Bitcoin[5] and the fork of Ethereum Classic from Ethereum[6].

For any blockchain algorithm, there are fundamental parameters that must be agreed on by all participants. If there is a divide on what those parameters should be, and this divide manifests itself in running blockchain software versions with different behaviors, the blockchain will fork, creating two new blockchains with different rules. This makes adjusting a blockchain's rules to changing needs a delicate topic, hindering its ability to evolve, and may instead lead to division and splintering, thus weakening the blockchain's ecosystem.

[5] Over the technical parameter of block size limit.

[6] Over the non-technical schism whether the smart contract involved in the DAO exploit was legal because it was accepted by the system, or illegal because it was unintended.

A key design goal of Dolla is to be more flexible and allow regular upgrades of the rules to meet new challenges. Thus Dolla was designed with a built-in method for evolution. The consensus members decide by voting not only on what transactions to include in the blockchain, but also on requests for changes to its own rules or voting members, and on whether or not to include new voting members. Immediate finality means that the whole system can upgrade atomically from one set of rules to another without risks of rule inconsistency[7].

Low cost

We aim to deliver a key promise of blockchain technology: To provide a reliable, trustable, decentralised, low cost mechanism for the transfer of value. The Dolla cryptocurrency provides an inexpensive alternative to conventional electronic payment methods, with equivalent levels of latency and throughput, and higher levels of transparency and resiliency due to decentralisation.

Transactions on the Dolla blockchain are executed at a small fraction of the cost compared to conventional payment methods. The target typical transaction fee at launch is expected to be set at approximately USD 0.01 (one cent) per two party transaction (2 outputs). This fee is almost insignificant compared to the expected average transaction amount, yet provides enough incentive for Dolla consortium members to keep their nodes operating and thus keep the Dolla blockchain running.

Providing an optimal base for smart contracts

Low latency and immediate finality provide an ideal base for useful, reliable and secure smart contracts applications.

While the first milestone of Dolla focuses on payments and does not include smart contract functionality, Dolla's design allows for first class support of smart contracts and work on smart contracts is already under way.

Privacy

For the first milestone, Dolla will offer privacy comparable with Bitcoin: All transactions with sender and recipient addresses are public, but anonymous as it is not clear which cryptocurrency users own which addresses.

We are currently actively exploring more sophisticated options (e.g. Ring signatures, Confidential Transactions, Stealth Addresses and zero-knowledge proofs like zk-SNARKS) which provide a higher level of privacy, protecting the identity of the sender and recipient, as well as hiding the transaction amounts.

[7] Because tying an upgrade to a specific block with agreed consensus is possible.

Formal verification and high quality implementation

Dolla aims to set new standards when it comes to correctness.

A mistake in cryptocurrency code or algorithms can easily lead to the theft or accidental loss of all coins involved. As a result, it is critical that both a cryptocurrency's logic and its implementation in code are without flaws.

In order of increasing correctness guarantees:

- Most cryptocurrency white papers present their algorithms and supply some argumentation on why they are correct.
- Some present mathematical proofs written and checked by humans.
- Few present a precise mathematical specification of their system.
- Even fewer present machine-verified proofs of correctness.
- None we are aware of present machine-verified proofs that the actual code implementing the algorithms is correct.

For Dolla we take the original [Cra+18] paper (which presents a human-checked proof) and strengthen it with a machine verification in Coq, a widely used theorem prover tool. The proof is extended to our amended version of DBFT, ADBFT, ensuring that our amendments are safe.

Dolla's reference implementation is written in Haskell, a purely functional programming language highly regarded for its ability to produce correct and safe software.

The code of the reference implementation is formally proven to be correct (that is, proven to implement ADBFT correctly) using the *hs-to-coq* tool, which allows taking Haskell code and embedding it in Coq proofs.

Using these high-assurance techniques, Dolla's users, developers, and third party auditors are provided with a new level of confidence that the cryptocurrency is safe and works as expected.

Furthermore, both Dolla's reference implementation code and proof code are thoroughly commented and they were developed using state-of-the-art engineering and QA processes, with a policy that all code must either be obvious or well-explained, so that it is easily scrutinised by the public and easy to contribute to.

Approach

Below we explain the ideas behind the design of Dolla, discussing

- a system overview of the components of the Dolla network,
- the consensus algorithm via which it is decided what blocks make it into the blockchain,
- how this consensus is used to implement a cryptocurrency ledger on which transactions move value between addresses,
- how the system is verified to be correct, and
- how the consortium that executes the consensus algorithm is governed and keeps Dolla running.

This chapter focuses on our general approach and design decisions. The next chapter gives detailed technical descriptions on specific points and analyse properties arising from the chosen design.

System overview

Dolla is a consortium blockchain and thus distinguishes between two sets of participants:

- **users**, which own monetary funds in anonymous accounts designated by *addresses*, and send and receive funds between addresses via *transactions*, and
- **consortium members**, which operate *consensus nodes* that maintain the network; they collect transaction requests from users, group them into *blocks* and collectively decide which blocks to include into the *blockchain*.

The number of users is unlimited.

The number of consortium members, n, is limited by the fact that the voting algorithm is expensive from a distributed systems perspective: To safeguard against consensus nodes failing or behaving maliciously (generally called "exhibiting *Byzantine* behaviour", see [LSP82]), every consensus node must relay any voting messages it receives from any node to every other node, an effort that grows asymptotically proportional to n for each individual node and n^2 for the entire network. The number of consortium members should be chosen as high as possible to tolerate more nodes being Byzantine, and as low as necessary so as to keep the cost from becoming prohibitive.

Users are anonymous, consortium members are explicitly not; their identities are public knowledge, they act with full transparency and are under public scrutiny.

The safety of the system does not depend on individual consortium members being trusted. It is proven to be safe and operating normally as long as less than one third of the nodes are Byzantine. It is a standard result of distributed systems theory that this threshold is optimal. Individual consortium member do not possess any power over the system. Any decision or statement is invalid unless supported by one third or more of

the members (which means that at least one honest member supports the decision; and it is proven that in this case, all honest members will support the decision).

The safety of Dolla depends on only one fundamental assumption: That less than one third of the consortium members collude[8].

Should one third or more of the consortium members collude, safety degrades significantly. However, it is possible to make it not degrade catastrophically. In no case can colluding members steal funds from the users, as that would require access to the users' cryptographic private keys which only the users have. However, users may still face loss of value by means of devaluation of the currency in the following scenarios: Should one third or more of the consortium members collude, they can certainly stop the network from processing transactions, withhold information from users, and deliver different information to different users, which may allow them to double-spend funds that they themselves own. However, such double-spends would likely not go undetected for long. The signed messages that nodes exchange would reveal which consortium members approved conflicting double-spend transactions, and because the identities of consortium members are public knowledge, they can expect to face repercussions[9]. The current implementation relies on the fundamental assumption stated above, and we are working on making detection of it being broken as easy as possible.

The set of consortium members is not fixed, it can vary over time. Anybody can apply to be a consortium member, and the consortium decides on it by majority vote. The same process may be used by the consortium to exclude badly behaving members. As these decisions are made based on vote counting, it is important that one can distinguish the nodes, so that every node can vote at most once. This is done via public-key cryptography, with a vote being bound to a node's private key. Further, the (personal or corporate) identities of any consortium member or applicant must be checked and published by the consortium. This ensures that they cannot apply as a new consortium member under a fake identity to obtain more than one vote[10]. The initial consortium is chosen so that the chance of a large number of members colluding is low (see section *Governance*), it is in the consortium's own interest to try to keep it that way when admitting new members.

Consortium members collect the fees that accrue, so it is important to have a simple

[8] This is the most important sentence in this document. Also note that all monetary systems rely on some fundamental assumption:

- Proof-of-Work systems generally assume that no set of colluding participants can achieve more than half of the computational power.
- Proof-of-Stake systems generally assume that no set of colluding participants can obtain more than half of the available stake.
- Consortium voting systems generally assume that less than one third of participants collude.
- Centralised systems assume that the central authority is trusted.

We believe that the consortium voting assumption is the most practical one, the most likely one to be detected should it be broken, and the one that degrades most gracefully.

[9] Note this is different in systems like Bitcoin: Since all participants are anonymous, an attacker who successfully executed a 51% attack to double-spend does not need to fear repercussions.

[10] This is a Sybil attack. Real-world identity verification is Dolla's protection against it.

incentive system that is designed to keep consortium members honest and working. They have an interest in keeping the system stable, trusted, and used by as many users as possible.

A user wishing to send funds to another user creates a transaction including the source addresses from which to spend the funds, the destination addresses to send the funds to, and a cryptographic proof that they own the private keys of the source addresses. The transaction is *submitted* to the consensus nodes as a *request for inclusion into a block* of the blockchain. A user creates, signs and send transactions using *wallet* software running on their device, that manages all addresses the user owns.

Dolla consensus executes in *block voting rounds*[11], also called *slots*[12]. In each such round, consensus nodes collect the transaction requests submitted to them by users. They combine all valid, not-yet-included transactions into a block, and *propose* the block to be appended to the blockchain. The nodes then vote which of the proposals is to be chosen. Exactly one proposed block wins the vote, and is appended to the blockchain. This process is irreversible, and the block and all transactions within are said to be *final* or *committed*. The voting round for the next block only starts after the current block is committed to the blockchain.

Because appending blocks happens in this lockstep fashion, the resulting blockchain is a single chain without any forks. The algorithm executed during one voting round is the ADBFT algorithm detailed in the next section.

Users can query consensus nodes for the blockchain contents, and thus discover whether a given transaction is in a block committed to the blockchain.

Consensus Algorithm

The Dolla blockchain is implemented using the algorithm from [Cra+18], with the following modifications, creating our Amended DBFT (short ADBFT) algorithm:

- Every decided proposal ("bin_decisions") is given an equal chance to be selected when a consensus voting round results in multiple acceptable proposals. This removes a bias inherent in the basic DBFT algorithm.
- The number of messages broadcast during consensus rounds is throttled by choosing one or multiple *slot leaders* for each slot. Non-slot-leaders are artificially delayed to increase the chances of slot leaders' blocks being accepted before non-slot-leaders execute their block broadcast. This reduces network message load because not each of the n consortium participants needs to propose a large number of transactions each slot.
- As a small improvement, non-valid blocks will not be echoed by any correct node.

[11] Note that one of the DBFT internal sub-algorithms also has a concept called "rounds". We use the terminology "block voting rounds" for the top-level rounds to distinguish the two.
[12] This terminology is used by some blockchains: "Blocks fill slots".

- A way to identify the original broadcaster of a block is added, by adding the ID of the proposer to the contents of the broadcast.
- Only the first proposal received from each node is considered (first `initial` message).

Below we first explain and elaborate some parts of the original DBFT algorithm[13], and then detail the above mentioned amendments.

DBFT terminology

When reading about DBFT, the following mapping of terminology may be of help:

Dolla terminology / specialisation	DBFT terminology
block	value
(consensus) node	process
a block is committed	a value is decided

Explanations of the original DBFT algorithm

The consensus procedure of [Cra+18] depends on four algorithms:

- MV-Propose - the final multi-value Byzantine consensus
- RB-Broadcast - reliable broadcast of a value
- bin_propose - binary Byzantine consensus
- BV-Broadcast - binary Byzantine voting with guarantees

In this section, we shortly introduce them, elaborate certain aspects, and provide justifications for logic that may not be immediately obvious.

RB-Broadcast

[Cra+18] uses the broadcast algorithm from [Bra87].

The objective of this algorithm is to send a message `m` to all (n many) participants of the network, such that all participants can be sure they agree on the contents of `m`, under the assumption that $t < n/3$ of the nodes are Byzantine. If the consensus algorithm relied on simple (non-"reliable") broadcasts instead, there would be no guarantee for the recipients that all other nodes in the network did in fact receive the same message, as N could deceptively choose to send different messages to each of them: the originator could be malicious and send different messages to different nodes.

[13] You may choose to read [Cra+18] before or afterwards, or skip it if short on time; however we do recommend you read it.

To mitigate this issue, "reliable" broadcast makes each node relay ("echo") received broadcasts to every other node, so that nodes giving different information to different nodes are ignored. Reliable broadcast is implemented in three messaging rounds, and there are three kinds of tags added to the messages to designate the round: `initial`, `echo` and `ready`. If a node wants to broadcast a message, it broadcasts that message together with the `initial` tag. Every node that receives a message tagged with `initial` re-broadcasts the same messaged tagged with `echo`. If a node receives a sufficient amount of agreeing messages tagged with `echo` that indicate that $t+1$ or more of the nodes agree on the value and are correct (this is guaranteed if more than $(n+t)/2$ `echo` messages have been received), that process can be certain that no other value could have been proposed. To finish the protocol, it broadcasts `ready` tagged messages containing the value, and it discards all further messages in regards to that broadcast. This is to ensure that only a single value will be accepted. Nodes will only accept the value if they received a sufficient amount of `ready` messages so they can be certain that every other correct process will accept the same value eventually. If this last step was left out, it would be possible that some node accepts a value while others wouldn't, even if they would never accept a different value.

Figure 1 shows an example execution of RB-Broadcast. The state of the nodes is shown on the inside of the nodes, where the first entry of the triple shows at which of the three stages the node is currently at, the second entry shows how many messages have been received after the round of messages, and the third entry shows what value has been accepted, if any. All messages in a single step are identical, and messages that nodes send to themselves are not drawn. The byzantine node in this example is unresponsive.

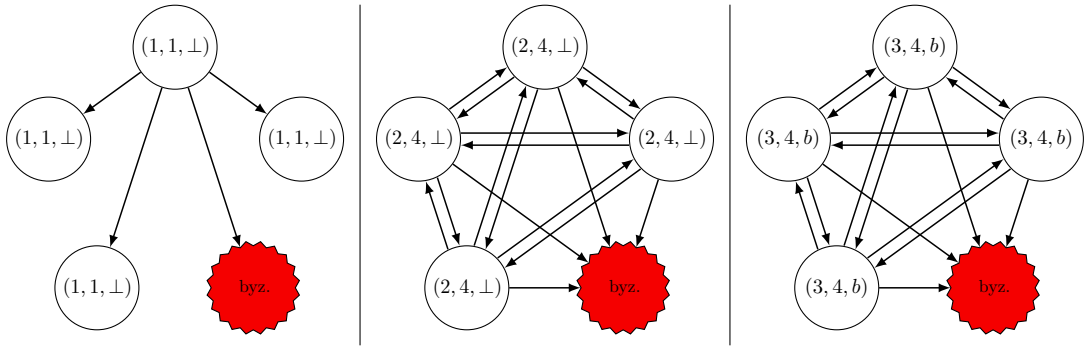

Figure 1: A node broadcasts `(initial, b, 1)`, then all non-byzantine nodes broadcast `(echo, b, 1)`. Finally, all non-byzantine nodes broadcast `(ready, b, 1)`.

This algorithm has the following important properties:

- If a value is accepted, the initiator sent the corresponding `initial` message to $t+1$ correct nodes. In particular, if the initiator is a correct process, it invoked `RB-Broadcast`.
- Every correct node will accept only a single value, and if a value is accepted, all correct nodes will agree on that value.

- If a correct node invokes `RB-Broadcast` with a value, eventually all correct nodes will accept that value.

BV-Broadcast

BV-Broadcast will broadcast a binary value in such a way that if the value is accepted by a correct process, there has to be a correct process that broadcast that value. An important distinction to RB-Broadcast is that multiple processes have to BV-Broadcast the same value to be able to accept that value. One can think of BV-Broadcast as trying to vote on a proposal, while RB-Broadcast is used to distribute information.

The algorithm is very simple: If a node wants to participate in BV-Broadcast, it broadcasts a value. If it receives a sufficient amount of broadcasts from other nodes for a certain value, so that the node can be sure that there has at least been one correct node that did broadcast that value, it re-broadcasts that value, but only if it didn't broadcast that value already. If it receives the amount of broadcasts necessary for it to know that every other process will also eventually accept the value, it accepts that value.

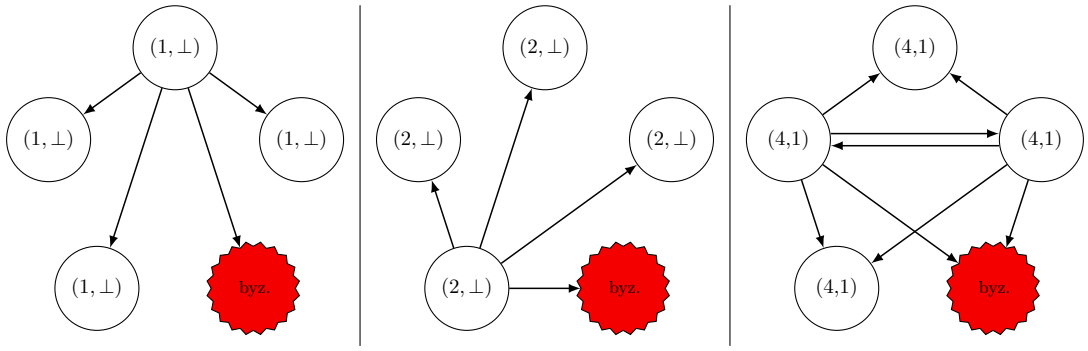

Figure 2: Two nodes propose the value 1. Afterwards, all the other nodes also broadcast the value 1.

Figure 2 shows an example of BV-Broadcast, with the broadcast of the value 1. The state of the nodes is displayed on the inside of the nodes, where the first element of the tuple is the number of received messages, and the second is the accepted value, if any, or bottom otherwise.

Note that it is possible that multiple values are accepted by BV-Broadcast, for example if all processes are correct and exactly half of them BV-Broadcast 0, and the other half 1. BV-Broadcast has the following properties:

- If a sufficient amount of processes invokes BV-Broadcast with the same value, it will eventually be accepted by all correct processes.
- If a correct process accepts a value, there exists a correct process that did broadcast that value.
- The set of accepted values of all correct processes will eventually become equal.

- If every correct process invokes BV-Broadcast, the set of accepted values will eventually become non-empty.

bin_propose

bin_propose is used to achieve consensus on binary values. As mentioned previously, `BV-Broadcast` may accept multiple values, which adds the issue that one can only be sure that no value gets added to the set of accepted values, if it already contains both possible values. bin_propose is therefore used to achieve *stable* consensus.

This algorithm uses voting rounds. Every message sent contains the round number in addition to what it intends to send. Also, the algorithm keeps an estimate of the eventual result, which is updated every round and which is initialized to a value that the node proposes.

At the start of each round, a node uses BV-Broadcast to propose its estimate. It then waits until BV-Broadcast accepts any value and broadcasts that value together with an `AUX` tag (as a regular broadcast). It then collects all messages with an `AUX` tag it receives until there are enough such messages such that 2/3rds of the nodes have to agree on the same values that have been accepted by BV-Broadcast. This will always happen after a finite amount of time.

If there are two possible such values, the estimate is updated to `current round number mod 2`. Otherwise, the estimate is updated to the single value. If that value also agrees with `current round number mod 2`, that value is accepted. If no value has been accepted, the round number is incremented and next round starts.

`bin_propose` has the following properties:

- All correct nodes that decide values will decide the same value.
- If all correct nodes propose the same value, that value will be decided.
- If a value is returned by `bin_propose`, there exists a correct process that proposed that value.

MV-Propose

This is the algorithm of the blockchain itself. It is a simple application of the previous algorithms. First, it invokes `RB-Broadcast` to propose a value. Then, as soon as any valid[14] value has been `RB-delivered`, it calls `bin_propose` with a `1` (yes) vote for that value, if `bin_propose` has not yet been invoked. Eventually, there will be some value where `bin_propose` returns a value of 1. When that happens, the node will invoke `bin_propose` on all other values with `0`. As soon as all instances of `bin_propose` have finished, the value proposed by the node with the smallest index that returned a value of `1` in `bin_propose` will be accepted.

[14] A value is valid in our setting if it is a correctly constructed block containing valid, non-conflicting transactions.

- All correct nodes will eventually agree on the results of `bin_propose` for all proposed values. Thus, all processes will return the same value (which has to be valid).
- There exists a correct process that invoked `bin_propose` for that value with `1`. Thus, that process had that value `RB-delivered`, which implies that all other correct processes will eventually have that value.
- All correct nodes will eventually decide on a value.

ADBFT - Amended DBFT (with our modifications)

This section describes in detail the amendments we have made to DBFT in order for it to be more suitable for Dolla. We call the resulting algorithm "ADBFT".

Removal of bias in the proposal selection

Because `MV-Propose` accepts the block from the node with the smallest index that has been voted `1` on, nodes with a smaller index have a higher probability of getting their block accepted. To work around this issue, instead of using the smallest index, we use a hash of `index + blockchain length` to determine the order of proposers. These numbers approximate an equal distribution, so for every block, every node has the same probability of having their block prioritized. Thus our replacement for line (08) of Figure 1 from [Cra+18] is:

$$(\_, \text{j}) \text{ <- } \mathbf{min}\{(\text{hash}(\text{x} + \text{numberOfBlocksInBlockchain}), \text{x}), \text{ such that bin\_decisions}_\text{i}[\text{x}] = 1\}$$

Where $(a_1, b_1) < (a_2, b_2)$ if and only if $a_1 < a_2$ or ($a_1 = a_2$ and $b_1 < b_2$), i.e. lexicographical ordering.

Addition of slot leaders

To reduce the load on the network infrastructure, in every slot some number of nodes are designated to be *slot leaders*. In their respective slots, the slot leaders' block proposals are heuristically preferred to be processed over other nodes. This is done by artificially delaying blocks by non-slot-leaders within the slot.

Note that this doesn't prevent any node from proposing outside of their slot, it only slows down their ability to do so. In fact it is critical that any node can continue to propose at any time so that Byzantine slot leaders cannot permanently halt the algorithm. In that case, the algorithm will simply take a bit longer to terminate. The safety guarantees of [Cra+18] remain unchanged.

The choice of number of slot leaders per slot is a tradeoff between network load and the probability of a slowdown introduced by all-Byzantine slot leaders and can be adjusted as needed.

Which node is leader in which slot is decided deterministically by hashing; every participant can compute this knowing only the consensus participants and the slot number (blockchain length).

Correct nodes that are not slot leaders voluntarily slow down the propagation of their blocks similar to the following operation:

operation Delayed_RB_broadcast(v_i):

if (i is not a slot leader) **then**
wait t **end if**;
if (not (exists l : bin_decisions$_i[l] = 1$)) **then**
RB_Broadcast VAL(v_i) **end if**.

Further, correct nodes will, in the implementation of `RB_broadcast`, delay echoing of messages originating from non-slot-leaders.

Block validity check

For efficiency reasons we start validation of each transaction upon receiving it, and the whole block is checked upon construction. The process can be started in parallel, and resulting mark is queried when the algorithm demands using `isValid` function.

Remembering the initial broadcaster of a block

In section 2.2 of [Cra+18], there is an operation `RB-deliver` mentioned that does not exist in [Bra87]. To have this notion of `RB-deliver`, one has to be able to derive the original broadcaster of a value, after a value is accepted. To solve this issue, the broadcaster is taken as part of the broadcast. Thus, if RB-broadcast accepts a value (k, v), we say that p_k `RB-delivered` v.

Ignoring of duplicate proposals

Careful inspection shows that the algorithm in [Bra87] does not satisfy `RB-Unicity` as required in section 2.2 of [Cra+18]. To comply with `RB-Unicity`, every `initial` message that arrives after a node already received an `initial` message from the same node will be ignored.

Transaction Processing

The logic of the consensus algorithm starts at the level of entire blocks. Below we describe how individual transactions interact with it.

Gathering transaction requests

As mentioned before, users submit transactions they wish to be executed to the consensus nodes. Concretely, a user wishing a transaction to be performed will submit it to as many consensus nodes as possible, in parallel. This is not strictly necessary, but reduces latency: It improves the chance that one of the receiving consensus nodes is a slot leader in the next slot, which can include the transaction in a block proposal immediately, and the chance that one of the receiving node is not Byzantine. Users with weak network connectivity may choose to submit the transaction to only one or a few consensus nodes, or probe nodes in sequence instead of in parallel, to trade latency in exchange for saving bandwidth. Latency for the transaction to be committed may be increased in this case because the node(s) may not be slot leader for a while and the transaction will be delayed to be proposed until that is the case, or because of the chosen node(s) being Byzantine and ignoring the request, in which case the user has to retry with a different node.

Consensus nodes, in parallel to executing the current consensus slot, collect transactions submitted to them by users. Consensus nodes should immediately reject transactions that are obviously invalid, such as the funds being referred to being nonexistent. At the beginning of the next slot, they bundle all received valid transactions in a block and propose the block to the other consensus nodes via ADBFT.

We have considered adding a transaction forwarding system, in which consensus nodes that are not slot leaders forward received transactions to those nodes that are (to further reduce the amount of bandwidth necessary for clients), but so far our benchmarks have not shown this to be necessary.

Checking transaction status

For a transaction to be committed, it is required that more than 1/3rd of the consensus nodes agrees that it is committed[15]. In that case, one can be certain that at least one correct node has accepted the transaction, which implies that all correct nodes eventually will agree on that transaction. Depending on the scenario, there are multiple possible approaches for a user to check whether a transaction is committed.

- **Checking by querying a quorum**: In this approach, a user asks as many consensus nodes as possible whether the transaction was included. As soon as 1/3rd of the nodes return a confirmation, the transaction is considered accepted.
- **Checking by using signatures**: In this approach, consensus nodes sign committed transactions (or rather, entire blocks that contain them), and present the relevant signatures (their own and those of other nodes) to users querying whether

[15]This is because less than one third of consensus nodes are assumed to be Byzantine. If one third of the nodes agrees that a transaction is committed, then at least one correct node agrees that it is committed. Correct nodes only do that if all other correct nodes agree on it. So one third of nodes agreeing in fact implies that two thirds will eventually agree.

a transaction was included. Being presented with signatures of 1/3rd of the nodes is sufficient for a user to know that the transaction was committed. Already contacting a single node that returns the required number of signatures is sufficient. These signatures can also be used to support the scenario where the recipient of funds is offline (but knows the public keys of the consensus nodes in the relevant time frame): The sender can present the signatures of the consensus nodes to the recipient. See the scenario Merchant is offline for details.

The second approach requires more functionality but allows a user to get the desired answer while contacting less consensus nodes, saving on latency and network traffic.

Formal Verification

This section describes the formal methods used to provide high assurances on correctness and security of the system.

The Coq theorem prover

Coq is a proof assistant which includes a dependently typed programming language. This means Coq can be used to state and prove mathematical theorems, a famous example of which is the proof of the four color theorem (see [Gon08]). Via *Gallina*, its integrated programming language, it can also be used to prove theorems about programs. Using the tool *hs-to-coq*, we can automatically translate our Haskell implementation of the Dolla blockchain to a Coq implementation, which we can then prove theorems about.

It is important to note that any Gallina code halts after a finite amount of time. This is enforced by the Coq compiler, which will only accept programs where termination can either be automatically verified by the compiler or where the user has proven the programs to terminate. As a consequence, by converting the code to Gallina and compiling it, it is already proven that it is impossible for the program to crash because of errors in the code. Crashes are possible only via external effects such as insufficient or faulty hardware (not enough memory, random byte flips in memory etc.), errors in the operating system, or similar.

Coq proofs on DBFT

As stated in [Cra+18], the general assumption is that at least 2/3rds of the consensus nodes behave correctly, i.e. follow the prescribed algorithm. Under this assumption, we are working on proving the following theorems from [Cra+18] with Coq:

- Any decided value is valid (Lemma 13 in [Cra+18]).
- Any two correct nodes will always agree on the values decided (Lemma 14 in [Cra+18]).

- The process to decide a value will terminate after a finite amount of time (Lemma 15 in [Cra+18]). Thus, it is impossible to attack the network via protocol operations in a way that it comes to a halt.

In addition to the above, it is possible to prove theorems not only about the consensus, but about the full implementation of the blockchain. This means that we can prove bounds on the maximum slowdown that a faulty consensus node can cause. We are working on proving these additional theorems with Coq:

- If a correct consensus node accepts a transaction, that transaction has been sent to it by a user that owns the correct private key (assuming signatures cannot be forged).
- If a user submits the same valid transaction to all correct consensus nodes and the network delays involved are sufficiently small with respect to timeouts, every correct node will eventually accept a block that contains this transaction.
- If a correct node is added to the system and the protocol for adding that node is initiated, the state of the node will eventually agree with the state of all other correct nodes.

Governance

Goals

The application process, staking requirements and deployment recommendations for consensus members discussed below are designed to promote the following results:

- decentralized network of serious, independent participants operating transparently and under public scrutiny
- geographically distributed network to support a global user base and to reduce the risk of being controlled by a single jurisdiction, political agenda, or fiscal plan
- diversity of types of consensus members (large commercial corporations, small businesses, nonprofits, educational institutions, individual experts etc.) to reduce the risk of collusion
- deployment across many different hosting solutions (private servers, cloud services, etc.) and technology stacks to reduce the risk of technical outages

Special consideration is given to applicants that contribute to the desired diversity of the Dolla network.

Governance before and after Handover

The above goals should remain in effect throughout the lifetime of Dolla. In the beginning, Lead One lays the corresponding foundation by selecting initial consensus members that are fit for the task and aligned with these goals. At some point post-launch, called

Handover, Lead One will move the control of Dolla to the consortium itself, and it will be the responsibility of the consortium to uphold these goals. Lead One will continue to issue recommendations on governance, but it is the quorum of consensus members that makes all decisions.

Consensus members

The Dolla network requires independent consensus members with the knowledge and resources needed to operate a node in a predetermined, safe and secure manner. The independence of each individual or organisation needs to be assessed and scrutinized to ensure that each member is truly a separate entity. To ensure transparency, the list of consensus members will be available to everyone, being under public scrutiny, and consensus members are encouraged to publish any relevant information about their peers. Each consortium member operates one node (which can consist of multiple machines). Each node is associated with one public key that identifies it.

You can read more about the responsibilities of consensus members in the section Roles and responsibilities in the Dolla network.

Requirements on consensus members for the initial consortium

These are the rules for applicants to the initial consortium selected by the Dolla founders. At some point after launch, the consortium may alter the rules, but it is expected that they will remain similar as they are tied to the goals and functioning of Dolla.

Identity and background checks

Applicants must show a proof of identity and disclose control structures and dependencies (such as organisation ownership, financial dependence, political associations). Such checks take a significant amount of effort. In case of a rejection, nothing is gained on the side of the Dolla founders (or later the consensus members). As a result, there will be a fee for applying as a consensus member, in order to fund these efforts.

Technical fitness

Applicants must demonstrate that they possess sufficient technical knowledge and resources in order to safely operate a consensus node. Applicants may also have to accept regular audits of their setups and security practices.

Legal contracts

Applicants may be asked to enter legally binding contracts (real-world, off-blockchain) to act according to the rules of the system. Most importantly, this includes that they

ensure that their node software acts as an honest, non-Byzantine node in the blockchain consensus.

Stake

Applicants, as well as admitted consortium members, will be asked that they hold a stake in the cryptocurrency, which acts as a security deposit. The willingness of a member to provide this deposit shows a certain level of commitment and discourages frivolous participation. The required stake amount will usually depend on the type of consensus member.

Diversity

The ratio of large commercial corporations, small businesses, nonprofits, educational institutions, individual experts and so on among the consensus members should be maintained as the number of members grows. The same applies to the distribution of members across different geographic regions, legislations and political systems. As a result, an otherwise suitable applicant from an overrepresented group may have to wait for diversity to increase by other members joining first.

Changing the rules

After the Handover event, binary voting ("yes" or "no") can be used to vote on acceptance of any proposed changes to the rules or extraordinary measures affecting the entire network. Proposals must be submitted by consensus members, either by developing them on their own or based on feedback from the community[16]. Proposals are typically first published and discussed off-blockchain. After a proposal has gathered enough interest, a human consensus member submits the proposal via their node software. All human members can then vote on that proposal via their node software. In practice, most proposals will be specialised smart contracts[17], so that all consensus members can atomically apply the changed rules. For example, one might propose to change a fee from amount A1 to amount A2, effective starting from block number B; if the proposal is accepted with a quorum as defined by the current rules, then all consensus nodes will apply the new fee when processing block number B. Accepting a new member into the consortium, expelling a member, changing existing parameters or changing the algorithms involved are all examples of rule change proposals.

The consensus node reference implementation will provide an additional convenience to turn algorithm changes into specialised smart contracts: It will feature an upgrade

[16] The Dolla founders will provide infrastructure, such as forums or Q&A sites, to foster a healthy discourse on how Dolla could be improved.

[17] "Specialised smart contract" here means special messages exchanged by the nodes and understood by the node software, which, when accepted, automatically change the behaviour of the software.

system, where nodes choosing to run the reference implementation can vote to atomically upgrade to a new version of the reference implementation.

The amount of voting power that a given member wields will depend on various factors such as invested stake, how long they have been part of the consensus, or how many blocks their node has contributed to process. However, voting powers of any single member must not be too large, in order to keep the risk of collusion low and the system decentralised.

Details and analyses

Preliminary Performance Evaluation

We use three approaches to estimate the eventual performance characteristics of the Dolla network:

- Theoretical analysis of the algorithms and data involved
- Simulation results
- Real-world benchmarks using geographically distributed test networks

As of writing, we are still working on generating more performance numbers. An upcoming revision of this paper will include further results.

We have implemented the ADBFT algorithm and evaluated its operation in a simulated network.

Our preliminary results show that under an assumed latency of 200 milliseconds between all nodes, a consortium of 40 consensus nodes can on average decide one value (block) within the necessary 1 second of time. Our benchmarks of cryptographic operations show the signature scheme of our choice can on current hardware perform more than the necessary 10,000 signature validations per second on a single CPU core. Correspondingly, we expect that the final Dolla network will achieve or exceed the goal of 10,000 TPS with ~1 second latency.

Note that the number of consensus nodes is limited by network throughput (when broadcasting the block to all members), not latency. We expect that we will be able to scale the number of nodes up significantly by using a more intelligent block distribution approach, such as a latency-guided tree scheme. We are currently working on simulations of such approaches that realistically model network throughput across geographically distributed nodes on the Internet. As a reminder, the number of Dolla users is independent of the number of consensus nodes. Even a small number of nodes can support millions of users. However, supporting large numbers of consensus nodes remains important to reduce the chance of collusion within the consortium and also reduces the risk of outage and improves decentralization.

Upcoming simulation results include:

- Geographically informed latencies
- How performance scales depending on consortium size, number of Byzantine nodes (down or deceitful)
- Effects of jitter, packet loss and network splits

In parallel to this we are working on benchmarks on the same topics in real-work test networks.

Comparison with other general approaches

In this section we compare Dolla's approach to cryptocurrency consensus with other general approaches. We do not introduce the approaches, but focus on their properties relevant to Dolla's goals instead. This provides a rationale for the approach chosen for Dolla. For direct comparisons with existing cryptocurrencies, see the next section.

Proof-of-Work (PoW)

It has been difficult for existing Proof-of-Work based blockchains to achieve transaction rates even above 100 transactions per second. Throughput in PoW is limited by *block time* (the expected time it takes to mine a new block) and *block size* (how much data, usually transactions, fits into one block). Latency in PoW is limited by block time, and the number of *confirmations* (number of blocks in the chain on top of a given block) that are assumed to make a block *reasonably certain* to be final. Since mining a block costs computational power, and the longest chain (the one with most work on it) is considered the source of truth, waiting for more confirmations reduces the risk that the system will switch to a different chain and one's block will be void. As a result, safety and low latency are antagonists in PoW: Waiting for less block confirmations reduces waiting time but increases the chance of a confirmed transaction being rolled back.

Even when using a high number of confirmations, the possibility of blockchain forks remains, so finality of the any transaction remains only probabilistic. In a recent attack against the PoW cryptocurrency Bitcoin Gold in 2018, high numbers of confirmations that had been believed to be safe were shown to not be high enough, as an 51% Attack brought up enough computational power to undo transactions that had been believed to be infeasible to reverse. As a mitigation against future attacks, the recommended number of confirmations was raised from one believed-to-be-safe limit to another believed-to-be-safer limit.

We believe that our goals regarding performance and safety mentioned above would be difficult, if not impossible, to achieve with a PoW blockchain.

PoW blockchains require an excessive amount of computing power and thus electrical power. For example, as of writing Bitcoin consumes an estimated 73 TWh per year, which exceeds the energy consumptions of large countries such as Chile and Austria. Reducing the amount of power needed often makes it easier for attackers to launch 51% attacks.

The Dolla blockchain uses the ADBFT consensus algorithm to achieve low latency and high throughput. It also does not permit forks, making any committed transaction final. The computational cost of executing the ADBFT consensus algorithm is a small fraction of the power needed to mine a PoW blockchain. The Dolla transaction fees are minimal (1 cent of a US dollar per typical 2 output transaction) and are used directly as incentives to reward active participation in building the blockchain.

Proof-of-Stake (PoS) and Delegated-Proof-of-Stake (DPoS)

Some blockchain implementations instead use Proof-of-Stake or Delegated-Proof-of-Stake consensus algorithms in order to achieve higher transaction throughput and avoid the high computational power cost of mining PoW blockchains. This has created an opposite problem by making it too easy to mine blocks (nothing-at-stake), so there is no disincentive to mine alternate forks as a way to disrupt the blockchain. In an attempt to prevent these abuses, these blockchains have resorted to non-trivial penalty schemes (i.e. slashing conditions) which bring their own set of problems and add complexity.

While PoS based blockchains have registered an improvement in transaction throughput over PoW blockchains, they still do not rival the transaction rates achieved by traditional payment methods.

In contrast, DPoS blockchains, theoretically can achieve much higher throughputs at the expense of decentralization. But they still suffer from the nothing-at-stake problem described above. Because these implementations do not eliminate forks, the resulting transaction confirmations remain non-deterministic and require more than one confirmation for the needed assurance.

By using the ADBFT algorithm, Dolla achieves the desired objectives in a simple and direct way, without any of these pitfalls.

Consortium

Having a consortium drastically simplifies the problem of consensus because it is known at all times what the participants of the system are. This makes it unnecessary to employ extra mechanisms to limit uncoordinated appending of blocks to the blockchain. Instead, one only needs an algorithm that reliably ensures that one of multiple possible block proposals is agreed on by everybody, which can be done by exchanging network messages and counting how many nodes agree with a block. This is a simpler and more efficient concept than mining, longest-chain rules and stake delegation methods. Making forks impossible is achieved naturally by simply running one block voting round after the other. Because the participants definitely know when a voting round is over and a block is accepted, the concept of "confirmations" as known from PoW/PoS/DAG is not needed, allowing you to achieve immediate finality and a very short block time (the time it takes to execute one block voting round). This makes consortium blockchains especially suitable for payment systems: Transactions are fast and vendors do not have to worry about choosing a number-of-confirmations to achieve a suitable balance between security and waiting time. A consortium allows a straightforward estimation of performance characteristics by considering how many members there are, how many message exchanges they need to perform for a block voting round, and how large the messages are. It also allows for accountability: Since the identity of all members are known, misbehaviour can more easily be detected and reacted upon.

Consortiums have two drawbacks: The need for identities and higher centralisation. It must be impossible to join a consortium anonymously, because that would permit the same individual to join the network multiple times under different identities and obtain disproportionate influence (a Sybil attack). To prevent this, it must be ensured that the consortium participants are distinct, usually by identity verification. As much as possible, it should also be ensured that participants are not controlled by the same entity (such as one participant bribing other ones) or colluding in common interest. Consortium blockchains are often more centralised than other approaches because not everybody can join by their own choice. However, this centralisation effect can be weakened by having the entire consortium itself perform the identity validations, as opposed to select members or external stewards.

Dolla is a consortium blockchain with vetted and publicly known members.

Comparison with existing cryptocurrencies

In this section we compare Dolla with some concrete alternative, existing cryptocurrency blockchains. For a more general discussion of blockchain approaches, see the previous section instead. All numbers reported are approximate and to our best knowledge at time of writing.

Crypto-currency	**Approach**	**Through-put** reported **TPS**	**Through-put** observed **TPS**	**Block time**	**Confir-mations** recom-mended	**Recommended total confirmation time**†	**Fees** ob-served **USD**	**Fees** peek **USD**
Bitcoin	PoW	7	3-6	10 m	6	60 m	1	55
Ethereum	PoW	15-20	8-16	15-30 s	30	6 m	0.2	5.5
Dash	PoW	48	25	2.6 m	6	15 m	0.2	1.6
Monero	PoW	4	0.05-0.13	2 m	15	30 m	1-2	20
Ripple	Consortium	1500	3-20	-	-	reported: 4 s observed: 2.3 m	> 0.01	0.42
Cardano	PoS	5-7	?	?	?	3-5 m	0.02	?
Dolla	Consortium	10000 *	tbd	1 s	-	1 s	0.01 *	tbd

Legend:

†	this is usually block time times recommended number of confirmations
*	planned for the launch of the currency
tbd	to be determined

Roles and responsibilities in the Dolla network

Users

Users may be consumers, merchants, or anybody who wants to own or transfer value (or in the future, execute smart contracts) via the Dolla cryptocurrency.

Goals and responsibilities:

- Install and run Dolla wallet software.
- Use it to send or receive Dolla.
- Keep software up to date.
- Keep wallet private keys secret.
- If able to do so, scrutinise or audit any code used in Dolla.
- Collectively boycott or fork the system should the consortium no longer act according to the original goals of Dolla.

Consortium members

Consortium members are individuals or organisations that keep the Dolla blockchain running. One member alone wields no power, but a quorum of members decides on all matters about Dolla via voting.

Responsibilities:

- Run Dolla consensus node software, on one or more servers.
- Keep all involved software up to date.
- Maintain the hardware and network infrastructure on which it runs.
- Continuously monitor and ensure the security of software, hardware and infrastructure under their control.
- Audit any code running in Dolla, including updates.
- Keep node private keys secret.
- Possess sufficient technical ability and resources to perform the above tasks appropriately.
- Scrutinise the behaviour of other consortium members and their software, and publicly report any relevant or alarming observations.
- Transparently disclose their own identities and other relevant information in the interest of public scrutiny.
- Propose new consensus members.
- Continuously monitor any topic that may indicate collusion of members or may increase the chances for future collusion.
- Perform or verify identity and background checks on new consensus members, as well as assess their technical fitness for this role.
- Vote on the admission of new consensus members.
- Propose changes to the algorithms and rules involved.

- Vote on changes proposed this way.

In return for their services, consortium members collect the transaction fees of the system.

Consensus node software

The node software is what runs on the servers maintained by consortium members. One Dolla node may consist of more than one server in order to provide extra computational or networking capacity.

Responsibilities:

- Receive user transactions through an API.
- Validate and accumulate transactions into a block.
- Propose a block to be added to the blockchain consensus.
- Maintain a local copy of the recent blockchain history, based on the outcomes of previous consensus rounds.
- Allow users and other node software to download recent blockchain history.
- Participate in voting rounds for blocks proposed by other consortium nodes.
- Notify consensus members of rule or membership change proposals that need human input for being voted on.

Dolla Team

This is the team which consists of Lead One team members and is responsible for the development of the Dolla transactional software (software required for the Dolla network to function).

Responsibilities:

- Fund the initial research and development.
- Design, develop reference software implementations, including wallet and node software.
- Perform simulations, measurements and studies to evaluate the impact of new ideas, rule changes and technical changes.
- Publish technical documentation and marketing material, including user manuals, consensus node manuals and best practices for consortium members.
- Train consortium members on how to best fulfill their roles and help with technical issues.
- Execute the initial funding event.
- Bootstrap the initial consortium, choosing members diversely in order to reduce the chance of collusion.
- Perform needed identity and background checks for consortium bootstrapping.

Transactions

The Dolla cryptocurrency does not have the notion of personal accounts like banks do. Instead, users may own and use Dolla coins without ever disclosing their identity to any third party. To accumulate coins in the network, users have access to *transaction outputs*; these outputs are stored in the blockchain, and they are referenced by an address. We plan to support multiple kinds of addresses in the future that enable different kinds of transactions. The following describes the base implementation of the cryptocurrency.

How transactions work

Coins in Dolla are always bound to an address that owns them. Yet ownership is not a simple mapping from address to coins owned. Instead, the system more fine-grainedly tracks all *transactions* of coins to and from a given address. When an address receives an amount of coins via a transaction, this amount is called a *Transaction Output (TxO)* which is said to be owned by the address. Until another transaction transfers it away from the address again, the amount is called an *Unspent Transaction Output (UTxO)*. An individual UTxO is indivisible and can only be spent as a whole; it cannot be partially spent. Thus, in order to transmit a certain amount of coins, one may have to spend a combination of multiple UTxOs, and likely receive some change back. This is similar to coins in a physical wallet, where individual cannot be divided. Each output has a unique ID in the blockchain. For the output to be spent as part of a transaction, the ID is written down in the transaction as an *Input*. This is where the "coins in a physical wallet" analogy ends: While two physical coins of same value are indistinguishable and can be used in place of each other, transaction outputs are distinguishable by their unique IDs. Further, physical coins have predetermined denominations (a fixed set of available values), transaction outputs can be of arbitrary value.

A Dolla Wallet creates a new transaction and submits it to one or more consensus nodes. Consensus nodes collect submitted transactions and bundle them into blocks. A block also carries a reference to the previous block in the chain. The consensus nodes execute the consensus algorithm to agree which block is to be committed (appended to the blockchain).

The consensus algorithm ensures that it is not possible to use the same committed output as an input of a new transaction more than once (does not allow double-spending). The consensus nodes verify this condition when a wallet proposes a new transaction.

Anatomy of a transaction

A transaction contains multiple inputs and outputs.

An output contains two key attributes: an address and an amount. The address is the

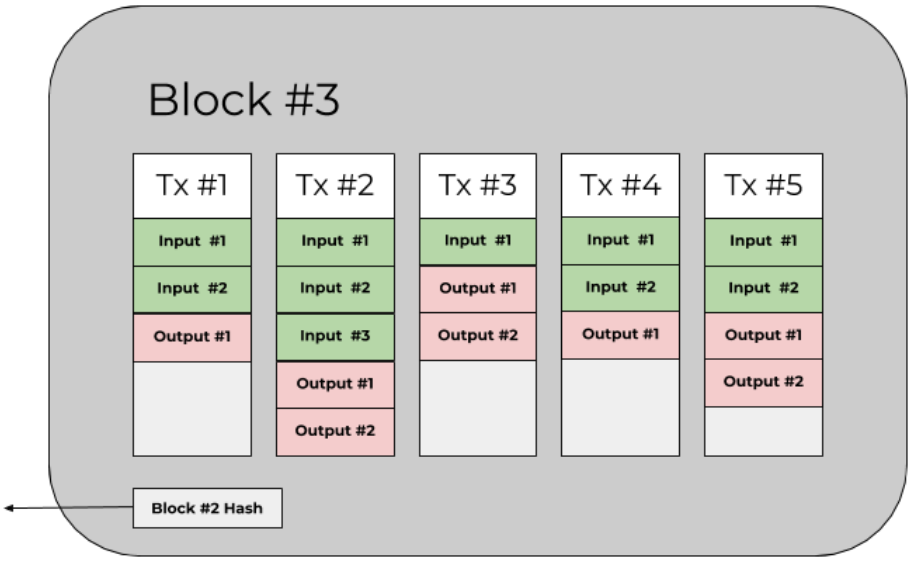

Figure 3: Transactions in a block

hashed public key of the recipient of the coins. This hashed public key serves as an identifier, similar to a bank account number to which third parties will transfer funds. The amount designates how much is transferred to the other party.

An input is a reference to an output from a previous transaction in the blockchain. A valid transaction must carry along signatures (one for each address that owns an input used in the transaction) that prove that the owner of the referenced output approves of it being spent.

The sums of input and output values in a transaction must be equal for the transaction to be valid.

Asymmetric cryptography

To secure the wealth that belongs to a transaction output, Dolla uses asymmetric cryptography, where users own cryptographic key pairs. One of the keys of this key pair is used by the sender to *sign* the transaction that contains an output that belongs to the sender; we refer to this key as the "signing key" or "private key". Only the owner of the output has access to this key. The other key of the key pair is used to *verify* signatures created that way, and is made public knowledge when an output is spent. This key is known as the "verification key" or "public key". A consensus node uses the verification

key to check that, in fact, the sender of the transaction owns the output that they want to spend.

Dolla relies on a standard elliptic curve cryptography scheme called Ed25519 (see [JL17]). This scheme offers some benefits (see [Ber+12]) compared to the other schemes like Secp256k1 (which most cryptocurrencies use, see [Qu99]). Specific benefits worth mentioning are:

- Foolproof keys: Ed25519 generates signatures deterministically. Key generation requires randomness, but signatures do not. This aspect can not only improve signing performance but is also easier to use safely when compared to Secp256k1, which, when not used carefully, can allow the derivation of private keys from public keys using the same random seed (the Sony PlayStation 3 security system is an example of where this went wrong).
- Small signing and verification key sizes: Ed25519 signatures are only 64 bytes and verification keys only 32 bytes large. Small key sizes are essential for minimizing the amount of storage and network traffic needed for the operation of the blockchain.
- No secret array indices: Ed25519 algorithms never read or write data from secret addresses in RAM; the pattern of addresses is entirely predictable. The scheme is therefore immune to cache-timing attacks, hyperthreading attacks, on leakage of addresses through the CPU cache.

The above are key reasons why we picked Ed25519 over alternative schemes for Dolla.

Hashing of recipient verification keys into output addresses

The address of a transaction output is a derivation from the verification key: a Blake2b-256 hash of it. Hashing the verification key of a transaction output adds the benefit of not exposing the verification key until the owner uses the transaction output as an input for a new transaction. This adds an extra level of security: If there were a vulnerability discovered on the Ed25519 scheme, it would not affect unspent addresses unless Blake2b-256 were broken simultaneously. That said, when the owner of an unspent output uses it as an input for a new transaction, the verification key is exposed, so that all participants can verify the legitimacy of the transaction. For this reason, it is discouraged to re-use the same output address more than once unless absolutely necessary.[18]

Mutual trust of a transaction's validity

It is an essential matter for merchants that sell services or goods to have guarantees that transactions cannot be reversed once a trade took place. Equally important is to be able to identify which goods are being paid for (or rather, which "bill" is being paid).

[18] One example where receiving multiple transactions to the same address is necessary is the concept of a "tipping jar" address, where generating a new address for everybody who wants to tip is infeasible.

Various scenarios are supported by the Dolla cryptocurrency while providing such guarantees. This works even when the participants lack Internet connectivity to the consensus network. In the next few sections we will use a common setup where Bob is a customer and is making a purchase from a merchant named Alice in some physical location, which for the sake of example is a bar, and demonstrate how payments can be made despite various degrees of online connectivity.

Scenario 1: Both, merchant and customer are online

Consider a situation where both parties have Internet connection and Bob, who is running his mobile Dolla wallet app, wants to buy a beer, which costs 1 Dolla. This happens via the following steps (steps 1-3 are the same for all scenarios described below):

1. Alice, the owner of the bar, generates a key pair and corresponding recipient address using her wallet.
2. Alice communicates the address to Bob. The actual means of a transfer is not relevant; it can be anything from near-field communication (NFC), scanning of a barcode to simple manual entry.
3. Bob's wallet, using a subset of his unspent outputs and Alice's newly created address, constructs a transaction that contains all of the necessary signatures. If all unspent outputs together amount to a larger value than 1 Dolla plus an associated fee, Bob's wallet will also add another output via which Bob receives the change due.
4. Bob's wallet submits the transaction to the consensus nodes and shortly after receives back an identifier describing where this new transaction is located within the blockchain.
5. Bob's wallet transfers the identifier to Alice's wallet.
6. Alice's wallet uses the identifier to query the consensus nodes to check if it really points to a committed transaction that moved the requested amount of 1 Dolla to Alice's address. If this is the case, Alice knows that Bob has successfully paid for the beer.

There might be a scenario when a merchant does not want to generate a new address for every transaction, but instead use a single address to receive all payments. In this case it becomes ambiguous who the current payer is. Then Bob could try to cheat in order to get a free beer by returning to Alice at step 5 an identifier for a transaction that was created around the same time by some other customer in the same bar who bought a beer for 1 Dolla. In order to prevent such mischief we can slightly extend the above process:

- In step 5, Bob's wallet also transfers some evidence that he owns the private keys in the transaction referred to by the identifier. This evidence could be a signature of a nonce that Alice's wallet additionally transfers to Bob's in in step 2.

Scenario 2: Merchant is offline

In the above scenario, both of the parties involved in the transaction had access to the Internet. However, this is not necessary for secure payment processing and validation. We now consider a situation where the merchant does have only sporadic Internet access and is only able to obtain the list of current[19] consensus nodes once every so often.

In this scenario, step 5 mentioned previously is no longer applicable since Alice cannot query the network of consensus nodes directly and is only able to get information about the transaction from Bob. However, since Alice does have occasional Internet access, she has a reasonably recent list of participating consensus nodes and their public keys. Even though Alice cannot trust Bob, he can still prove to her that the transaction submitted at step 4 has been committed by asking the consensus nodes to provide him with signatures of the block that includes the transaction, and presenting the block and signatures to Alice. This requires the *Checking by using signatures* scheme described in Checking transaction status. Then concretely:

- In step 4, Bob's wallet obtains the block including the transaction, including consensus node signatures.
- Instead of step 5, Bob's wallet transfers it to Alice's wallet.
- Instead of step 6, Alice's wallet would check whether enough block signatures of the consensus nodes she knows about are present.

If Alice's list of nodes is too out-of-date, she will either reject the transaction, as Bob may not be providing signatures of the nodes she expected, or she may be more vulnerable to consortium member collusion if the consortium has grown substantially since her last sync, in which case her wallet may accept the transaction by being presented with less than 1/3rd of the *current* members' signatures. In practice, this is unlikely to be a problem, as growing the consensus network requires human involvement and is a time intensive process, and even users with sporadic Internet access will synchronise quite often in comparison. The safety of the approach can be further increased by establishing a consensus rule that the set of consortium members can only change by a certain fraction over a certain amount of time.

Scenario 3: Customer is offline

With the situation reversed, when the merchant does have Internet but the customer does not, their responsibilities get reversed as well. In this case:

- In step 4, Bob's wallet does not submit the transaction to the consensus nodes, but instead transfers it to Alice's wallet, which submits it to the consensus nodes.
- Alice's wallet performs all remaining verification steps; Bob is no longer involved.

[19] Remember that the consortium can change over time as members are added or expelled.

This approach works fine until we take into account the possibility of a failure. In particular, the transaction constructed by Bob and submitted by Alice can get rejected by the Dolla network because of some validation error (for example, Bob's wallet software may be out of date). In that case, Bob may be unsure whether there was an actual error on his side or whether Alice is pretending that Bob ran out of money, while silently keeping his coins. Bob will be able to discover the truth only once he has Internet connectivity again, at which point his wallet can inspect the blockchain in order to determine whether Alice did execute the transaction or not. If he finds she didn't, Bob should at that point transfer the involved funds to a newly generated address he himself owns, so that Alice can't claim them later. If he finds she did, he should complain to Alice that he paid but did not receive his beer. In general, Bob should opt in to this approach only if he reasonably trusts Alice, or if he knows that he'll have the opportunity to return later and complain.

Scenario 4: Both customer and merchant are offline

With both parties lacking access to the Internet and the ability to communicate with the network of consensus nodes, all security guarantees are gone. If Alice and Bob trust each other reasonably (for example if Alice is a reputable merchant and Bob is a known customer, or if they know each others' identities and can litigate a fraud), they may agree to create an offline transaction and either one of them would submit it to the network once online. But technically there is nothing that prevents Bob from spending the coins before Alice can submit it to the network, if Bob comes online first. The only guarantee is that once any party is online again, they can verify using the blockchain whether or not a transaction was made.

Fees

The motivation behind transaction fees is to provide an incentive for consortium members to invest computing and infrastructure resources in order to actively participate in building and maintaining the blockchain. In detail:

- The fees are structured so as to keep the cost for the average user at a minimum
- Transaction fees are only charged per output. A typical transaction can have any number of inputs, but usually has only two outputs, one output as payment to the recipient, and one output as change back to the sender.
- Initially, the fee per transaction output will be set so that the cost of a typical two output transaction is approximately equivalent to 1 cent USD. This can be adjusted via a rule change consensus vote of the consortium members.
- Initially, fees will be distributed amongst all consensus nodes. The exact distribution rules may depend on invested stake or other properties.
- Under this simplified transaction fee structure, non-participating or improperly maintained consensus nodes can be penalized by being temporarily dropped from the fee distribution, through consensus vote of the consortium members.

In future versions, the fee structure can be modified, at the discretion of the consortium, to:

- provide varying incentives for different aspects of consensus node participation (i.e. collecting user transactions, proposing blocks, voting, etc.)
- penalize non-participating or misbehaving nodes, even in proportion to the severity of their misbehavior
- provide incentives to users for transactions that are beneficial to the blockchain (i.e. transactions that consolidate a large number of smaller inputs).

Example of a transaction being processed

To facilitate understanding, in this section we give an illustrated example of how a typical transaction is processed in Dolla.

All values involved are arbitrarily chosen example values.

1. A customer (payer) wishes to purchase a good from a merchant (recipient) in exchange for 8 Dolla coins.
2. The recipient creates an address where they expect the 8 coins to arrive and communicates it to the payer.
3. The payer creates a new transaction, which includes:
    a. Transaction inputs (references to unspent outputs), the sum of which matches or exceeds the amount that the ayer expects. In this example, this is one unspent output (UTXO) containing 10 coins.
    b. Signatures that prove that the payer owns these unspent outputs.
    c. A transaction output valued 8 coins to the recipient's address.
    d. A transaction output valued 1.9 coins as the change.
    e. The transaction fee (0.1 coins in this example).

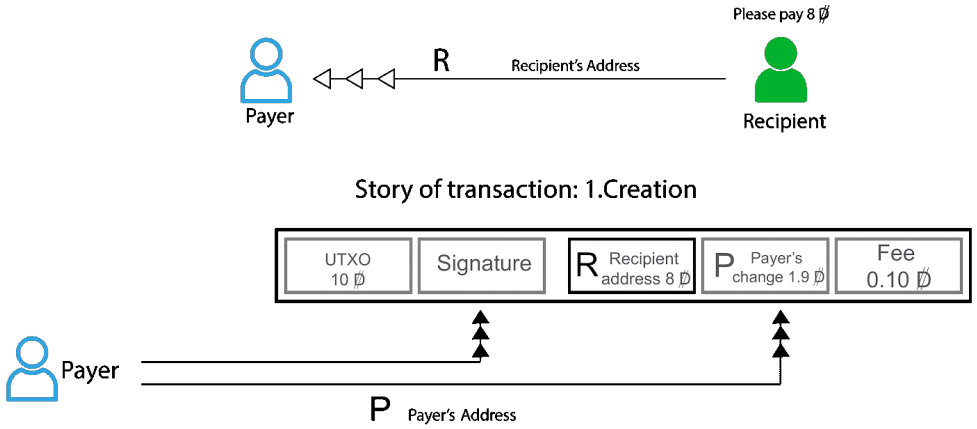

Figure 4: A payer creates a new transaction to a recipient's address

4. The payer sends the signed transaction to one or many of the consensus nodes, through an API served by each node.
5. The consensus nodes check the validity of the transaction by checking the transaction's signed inputs against unspent transactions in the blockchain and also against duplicate transactions within the new block being built (double-spend attempts).
6. Each consensus node accumulates multiple valid transactions into a block to propose.

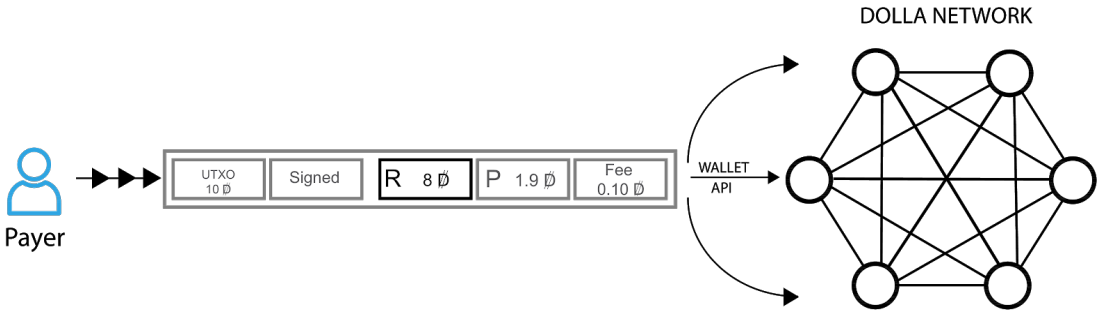

Figure 5: The payer submits the transaction to the consensus nodes

7. The consensus nodes execute the ADBFT consensus algorithm:
    a. Each consensus node proposes their block by broadcasting the new block to all other consensus nodes.
    b. To assure the integrity of the broadcasted value each node also re-broadcasts (echos) the proposed values that it received to every other consensus node in the network.
    c. Once the broadcasted value is deemed reliable (having been confirmed by a sufficient number of nodes), each node checks the validity of the proposed block and votes for or against the proposal by broadcasting its decision to all consensus nodes.
    d. To assure voting integrity, each Dolla consensus node also re-broadcasts (echos) votes received, to all of the other Dolla consensus nodes.
8. Once consensus is reached on a particular proposed block, each Dolla consensus node adds the block to its local copy of the blockchain.
9. The recipient is informed of the successful transaction using one of two common approaches (see Checking transaction status):
    a. The payer retrieves the block from the blockchain and presents it to the recipient.
    b. The recipient themselves retrieves the block from the blockchain.

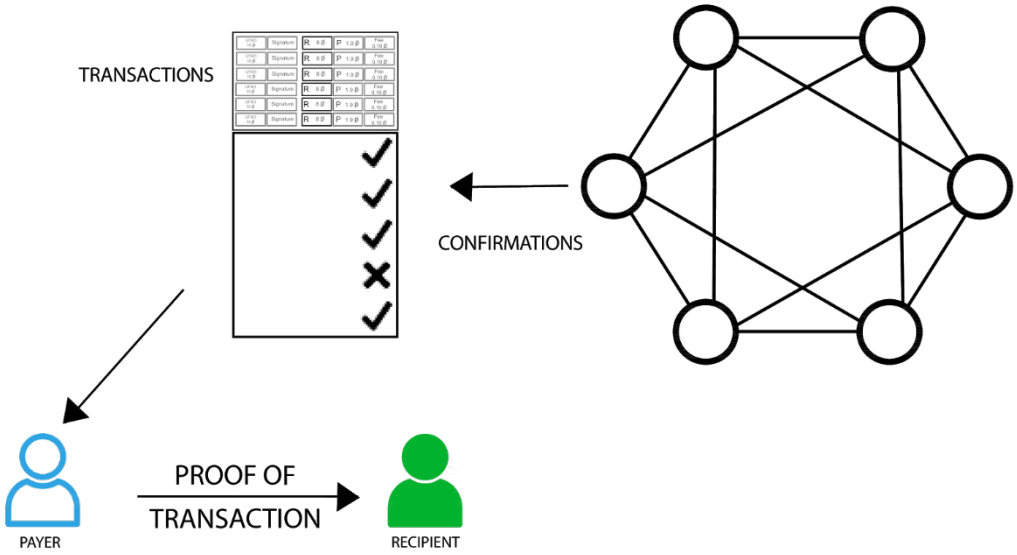

Figure 6: The payer retrieves a finished transaction and presents it to the recipient

Consideration of possible attacks

In this section we discuss a range of common attacks against blockchains and discuss their relevance for Dolla.

Double-Spend

A double-spend is when a user successfully fools the system into allowing them to spend the coins that they own more than once. Double-spends are not attacks, but are often an outcome of an attack.

Relevance for our system: The Dolla blockchain prevents double-spending by design. The absence of forks and the property of immediate finality means that it can always be determined clearly what money is spent and what money is unspent.

Majority Attack (aka. 51% Attack or >50% Attack)

In Proof-of-Work:

- An attacker spends coins on the blockchain in exchange for goods, services or other valuables. Simultaneously, the attacker expends computing power to create a fork in the blockchain and grow it as long as possible. If they succeed to make their alternate fork longer than the main blockchain that contains spending of the

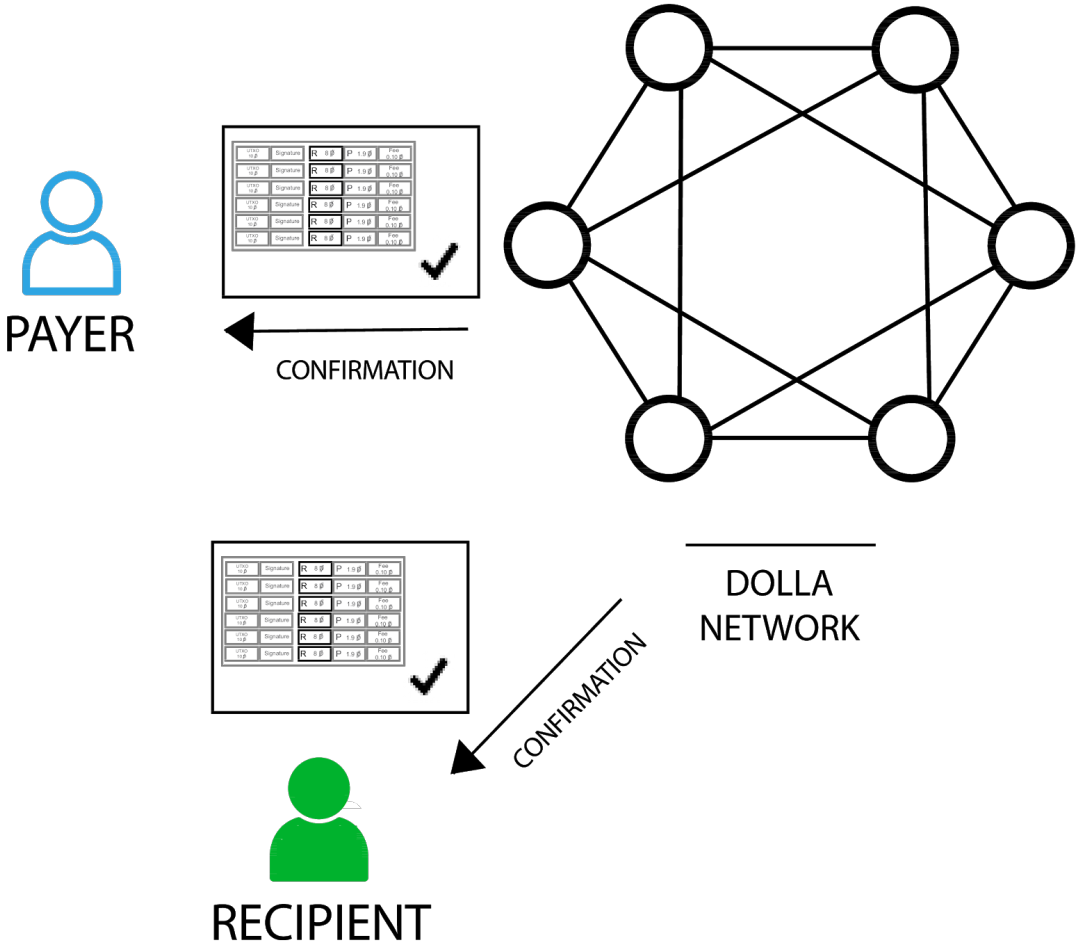

Figure 7: Both payer and recipient retrieves a completed transaction

attacker's coin, the system will to switch to the alternate fork instead. Then the attacker will regain the spent coins and can spend them again (a double spend), and whoever was paid by the coin will lose them.

In Proof-of-Stake:

- An attacker gains the majority of stake and thus voting rights, allowing them to let double-spends pass through.

In consortium blockchains: An attacker manages to make $t = n/3$ or more of the n consensus members collude in their favour, allowing them to let double-spends pass through consensus votes.

Relevance for our system: Since Dolla is not a Proof-of-Work or Proof-of-Stake blockchain, the corresponding 51% attacks do not apply to it. An attacker managing to make sufficient number of nodes collude is a threat to Dolla.

It is addressed by

- selecting the initial consortium such that collusion is unlikely and enforcing a degree of transparency so high that collusion is likely to be detected (see section Governance),
- making the penalties for collusion severe, and
- technical means (signatures) that make whistle-blowing on misbehaving members easy.

Sybil Attack

An attacker runs or impersonates multiple impostor network nodes to pose as real nodes in order to gain undue influence over the network.

Proof-of-Work and Proof-of-Stake approaches are usually not susceptible to Sybil attacks, since splitting one's node doesn't provide more computational power or owned stake.

Relevance for our system: Dolla requires identity and background checks for consortium members, as well as a high degree of transparency, in order to reduce the chance for Sybil attacks.

Eclipse Attack

This attack is against blockchains that use peer-to-peer network level gossip.

An attacker manages, using various techniques, to monopolize all of the peer-to-peer connections to and from a victim node, effectively isolating it from the rest of the network. This allows the attacker to control which other nodes the victim node can or cannot communicate with, and thus to withhold information that the security of the system relies on.

This crippled network state can allow and facilitate other types of attacks with less resources than would otherwise be needed. For example, it can facilitate a successful Majority Attack, even if the attacker controls less than 50% of the nodes.[20] It can also facilitate a double-spending attack, by allowing the attacker to split his duplicate transactions between the isolated node's view of the blockchain and the rest of the network.

Relevance for our system: In Dolla, all consensus nodes are publicly known. This includes their public keys, which makes it easy to identify which nodes a node is talking to, and makes it impossible for an attacker to impersonate other nodes. Dolla's consensus algorithm does not become unsafe when connectivity is lost; instead it stalls until connectivity is restored. This makes Eclipse attacks irrelevant for Dolla, as increased isolation cannot trick a node into making a worse decision.

Long Range Attack (aka. Alternative History, History Revision)

Similar to an 51% attack, but the attacker creates an alternate blockchain starting from the Genesis Block (the 1st block in the real blockchain). In other words the attacker forks the blockchain from the very beginning in an attempt to make his alternate blockchain overtake the real one, eventually replacing the real blockchain with his. This is done by exploiting weaknesses specific to Proof-of-Stake consensus algorithms; see more about it here.

Relevance for our system: Since the Dolla blockchain cannot not have any forks, this attack is not relevant to it.

Transaction Flood Attack

An attacker submits to the blockchain as many valid transactions as he can, to and from multiple accounts that he controls. The intention is to cripple the blockchain's network by saturating it.

Relevance for our system: Dolla is designed to handle a large number of transactions per second. Consensus nodes can easily be scaled to multiple machines to handle higher load. Further, while the fees for individual transactions are low, creating a flood of valid transactions would still be forbiddingly pricy. As an example, at a cost of ~1c USD per transaction, a 10000 transactions per second attack would cost $100 worth of fees per second. Finally, consensus nodes have the option to raise fees via a rule change to mitigate this type of attack.

[20] For an illustrated example, see https://medium.com/mit-security-seminar/eclipse-attacks-on-bitcoin-s-peer-to-peer-network-e0da797302c2.

Man-in-the-middle Attack

An attacker inserts himself into the unencrypted/unauthenticated communication paths between two network nodes and is able to impersonate each one to the other. his effectively allows the attacker to intercept, modify or replace any message sent between the nodes.

Relevance for our system: Dolla being a consortium system makes it trivial to not have any unencrypted/unauthenticated communication paths, rendering this attack impossible.

Network-level Denial of Service (DoS) Attack

This form of attack is similar to a Transaction Flood attack, except that the attacker floods the victim node with network packets that are not necessarily valid from any perspective. In a Distributed DoS (DDoS) attack, the attacker does this from multiple distributed sources, making it difficult to simply block or drop the traffic. This type of attack can effectively incapacitate or severely cripple the victim node from participating in network communications.

All systems on the open Internet, including Dolla, are susceptible to this type of attack. It is best addressed by matching and exceeding the attacker's network resources. This requires that the design of the system enables it to easily and temporarily scale up the amount of machines and networking power involved.

Relevance for our system: Dolla's decentralised nature means that an attacker must hit many consensus nodes at once in order to achieve any effect. In case they succeed, the cryptocurrency is guaranteed to only be slowed down, not to become unsafe. Finally, Dolla's design makes scaling up individual consensus nodes to multiple machines easy, allowing node operators to match an attacker's resources.

Desynchronization Attack

An attacker prevents the victim node from free access to mechanisms designed to keep all the nodes on the network properly synchronized (i.e. a time server). Desynchronization can also come about as a result of degraded network conditions, causing abnormal message delays, not necessarily due to an intentional attack. Either way, if a system relies on synchronisation in order to function, this can cause a desynchronized node to behave erratically or unreliably.

Relevance for our system: Dolla's safety does not depend on synchrony or access to accurate time, so it is immune to this type of attack.

The Nothing At Stake Problem

In Proof-of-Stake blockchains with forks, where mining doesn't cost significant resources, it is an optimal strategy for anybody to mine on both sides of a fork, as there is no cost to it and each side offers potential rewards. Adversaries can exploit this to more easily create forks, since even when the attacker owns only a small amount of stake, other participants are likely to support the attacker's fork to reap the potential rewards.

Adversaries, and more specifically past stakeholders who no longer have a stake in the blockchain can exploit old blocks to build an alternate blockchain to their advantage.

Relevance for our system: Dolla is not a Proof-of-Stake blockchain with forks and thus not affected by this problem.

Transaction denial attack

An attacker tries to prevent a specific transaction from being committed to the blockchain (for example, to stop a specific user from spending their funds).

Relevance for our system: In Dolla, only consensus nodes decide on what transactions make it into the blockchain. Should a user observe that a specific consensus node is not acting upon their request to include their transaction into a block proposal, they can simply submit their transaction to a different consensus node.

Glossary of terms

Binary Consensus (bin_propose) bin_propose is used in the ADBFT consensus algorithm to let the nodes agree on a binary value, with the guarantee that the decided value was proposed by a correct node.

Binary Value Broadcast (BV-Broadcast) Algorithm used by bin_propose to broadcast a proposal to the entire network, with the guarantee that an accepted value was proposed by a correct node.

Block The basic element used to build the blockchain. It mainly contains committed transactions, submitted by users and accepted as valid by the consortium members.

Blockchain A data structure, made up from blocks of data chained together by each block including the hash of the previous block. Exact copies of the blockchain are simultaneously maintained by multiple distributed nodes which assures its integrity.

Block Time The time it takes to add a new block to the blockchain.

Block Voting Round (also called Slot) A consensus voting round of the consensus nodes to decide on which proposed block should be included in the blockchain. This term is used to differentiate it from the lower level bin_propose voting rounds.

Byzantine Fault Tolerance A characteristic of a distributed computer system that allows it to continue operating (fulfilling its function) in the presence of a tolerable level of node failures or arbitrary behaviour, and even when the failure state of nodes cannot be determined with assurance.

Byzantine Node A network node that is faulty, malicious or behaving arbitrarily.

Byzantine Consensus Consensus reached by consortium members on a proposed value, even in the presence of Byzantine nodes.

Committed Transaction A transaction that has been registered on the blockchain, meaning that it is in a block that was successfully added to the blockchain. Once a transaction is in the Dolla blockchain it can not be removed or changed.

Consensus Algorithm The message protocol used by the Dolla network consensus nodes to reach agreement on a proposed value. The value is usually a new block to add to the blockchain.

Consortium The group of entities that participate in the consensus protocol of the blockchain. Each member is represented by a Dolla consensus node in the blockchain network and votes for proposals through its node.

Consortium Member An entity that is part of the blockchain's consortium and therefore can both propose new values or vote for values proposed by other consortium members. Runs consensus node software.

Coq A programmatic tool used to state and prove mathematical theorems. It can also be used to prove theorems about computer programs.

Decide a Block The principal goal and conclusion of a blockchain consensus protocol is to agree on the disposition of a new block proposed by one of the consortium members. Blocks that were agreed on ("decided") are appended to the blockchain.

Delegated Proof-of-Stake (DPoS) An optimization of the Proof-of-Stake consensus algorithm, whereby stakeholders delegate their votes, in proportion to the size of

their stake, to a small number of actual voters. This can allow the blockchain to operate with higher transaction throughput.

Double Spending When an owner of coins registered on the blockchain (unspent outputs) succeeds in fooling the network into allowing them to spend their coins more than once (i.e. use the same coin to make two different payments or transfers).

Fork When a blockchain protocol allows for multiple versions of a blockchain to exist simultaneously, usually going forward from a certain block. At some point one version is chosen as the "real" one, thus making the other version(s) invalid.

Formal Verification Proving the correctness of an algorithm or program against its specification using formal mathematical methods.

Functional Programming A programming paradigm where computer programs are built by composing smaller functions into larger ones. It is characterized by immutability of values, referential transparency of functions, the strict control of side effects, and strong type discipline.

Governance The set of rules and policies that determine the initiation, management and execution of changes to the Dolla blockchain.

Haskell A general purpose purely functional programming language. Due to its purity (ability to guarantee absence of side effects) it is well suited for writing high assurance code and for being used in formal verification via automated proofs. It is the language chosen to build the reference implementation of the Dolla blockchain code.

hs-to-coq An automated tool that converts Haskell programs into Coq code in order to facilitate formal machine verification.

Latency Generally, it is the time between a request and the corresponding response. More specifically, for the Dolla blockchain, it is the time that it takes for a transaction to be committed.

Peer-to-Peer A method of communication between nodes on a network that uses a direct channel between the communicating nodes, as opposed to communicating through a server.

Public/Private Key Pairs As used by the Dolla blockchain, this refers to an asymmetric cryptographic key pair generated using an elliptic curve algorithm (Ed25519). The private key is known only to the owner, while the public key can be revealed to anyone.

Proof-of-Stake (PoS) Consensus algorithm where the opportunity to add a block to the blockchain is granted randomly to a subset of the participating entities, weighted by the total amount of coins that the entity owns (the size of the entity's stake).

Proof-of-Work (PoW) Consensus algorithm where the opportunity to add a block to the blockchain is granted to the first entity (miner) that solves a computationally expensive problem. It is common to regularly increase the difficulty of the problem to be solved, as more miners add hashing power, which leads to diminishing incomes for the miners, consequently increasing the amount of energy burnt by such networks.

Proposed Block A block proposed by a Dolla consensus node to be added to the blockchain. This block contains all valid transactions received and accumulated by

the proposing node since the previous block was added.

Reliable Broadcast (RB-Broadcast) A broadcast algorithm used in the ADBFT consensus algorithm to reliably broadcast values (usually blocks) to the rest of the network, which tolerates a certain amount of Byzantine nodes.

Signature (digital signature) A piece of data that can only be created using a certain private key. It can be verified whether such a message was created with that private key by verifying the signature with the associated public key. In the context of the Dolla blockchain, some uses of signatures are to prove ownership of coins, authenticate nodes, etc.

Transaction An exchange of blockchain coins between two parties. It is created and signed by the sender of the coins, and then submitted to the network for inclusion in the blockchain. Once the transaction is part of the blockchain it is considered committed and the transferred coins are then available for the recipient to spend.

Transaction Inputs References to coins on the blockchain (unspent outputs), belonging to the sender, and being transferred to the recipient, possibly with some change going back to the sender.

Transaction Outputs Specifies the amount and destination address of a transfer or payment. Until these outputs are spent (transferred further to another address), they are called Unspent Transaction Outputs.

User Anyone that owns coins on the blockchain.

Valid Transaction A well formed transaction whose inputs are verified to be unspent transaction outputs and whose signatures prove that the creator of the transaction owns the funds referred to within.

Wallet Client software that tracks the coins owned by a user and allows a user to initiate a signed transaction. It includes tools to generate and manage the user's public/private key pairs.

A Survey of Attacks on Cryptocurrencies

In this section we have curated a list of recent attacks on various cryptocurrencies. It is intended mainly to save the reader some research and give some insights on what can go wrong with cryptocurrencies, and how much value can be lost as a result.

The reader is encouraged to consider these past attacks to evaluate the design of Dolla.

Attacks by category

51% Attacks

Perhaps the most fundamental type of attack on various blockchain types. See Majority Attack for a description of this attack.

- August 2018: a Brief History of 51% Attacks
- June 2018: Zen Cash 51% Attack ($700,000)
- May 2018: Bitcoin Gold 51% Attack ($17.5 Million)
- May 2018: Litecoin Cash 51% Attack
- April and May 2018: Verge 51% Attack ($2.85 Million)

Mining Malware

The proliferation of blockchains that are vulnerable to 51% attacks, and the recent success of such attacks, have also created a huge incentive for the creation of aggressive malware that hijacks the computing power of unsuspecting users for the benefit of the attackers. his is an insidious threat to the public as a whole and not only to those that participate or trade in these currencies. By making 51% attacks impossible, the Dolla blockchain is an alternative that, by design, defuses these corrosive incentives.

- June 2018: Cryptocurrency Mining Malware
- 2017: Anti-Phishing Working Group Report

Attacks Exploiting Blockchain Implementation Bugs

Even simple bugs in the implementation of a blockchain can easily short circuit the most robust security strategies built into its design and specifications. The approach used to design and develop the Dolla blockchain was chosen to proactively mitigate these risks (for details please refer to Formal verification and high quality implementation).

- September 2018: Possible Double-spend in Bitcoin, Bitcoin ABC and Bitcoin Unlimited
- March 2014: Mt. Gox Hack ($473 Million)

- August 2010: Bitcoin Hack (184 Billion Bitcoin)

Denial of Service Attacks

See Denial of Service Attack for a description of this attack.

- June 2018: Bitfinex DDoS Attacks

Wallet Attacks

These attacks are against the wallet software running either on the users' machines or as web services.

- July 2018: Bancor Hacked ($13.5 Million)
- June 2018: Shopin Hacked ($10 Million)
- January 2018: Coincheck Hacked ($530 Million)
- November 2017: CryptoShuffler Trojan Steals Cryptocurrency ($140,000)
- November 2017: Parity Wallet Bug ($280 Million Frozen)
- July 2017: Parity Multi-Sig Wallet Hack ($32 Million)
- April 2017: Yapizon Hacked ($5 Million)
- August 2016: Bitfinex Hacked ($72 Million)

Smart Contract Attacks

In these attacks, smart contracts were exploited to achieve unintended transfers.

- July 2018: KICKICO Hacked ($7.7 Million)
- July 2018: Bancor Hacked ($13.5 Million)
- November 2017: Parity Wallet Bug ($280 Million Frozen)
- June 2016: DAO Hacked ($50 Million)

ICO Attacks

In these attacks, cryptocurrencies were exploited already during their ICO phase.

- July 2018: KICKICO Hacked ($7.7 Million)
- August 2017: Enigma ICO Hacked ($500,000)
- July 2017: CoinDash ICO Hacked ($7 Million)
- July 2017: Veritaseum ICO Hacked ($8 Million)

Other Attacks

For these attacks, the exact vulnerability or other details were not revealed.

- June 2018: Bithumb Hacked ($31.5 Million, second time)
- June 2018: Conrail Hacked ($40 Million)
- May 2018: Monacoin Selfish Mining Attack ($90,000)
- February 2018: Bitgrail Hacked ($195 Million)
- December 2017: NiceHash Hacked ($70 Million)
- November 2017: Tether Hacked ($30 Million)
- July 2017: Bithump Hacked (first hack)
- July 2016: Steemit Hacked ($85,000)
- January 2015: Bitstamp Hacked ($5 Million)

Attacks in chronological order

The following list of attacks is in reverse chronological order, specifying the attack vectors or vulnerability exploited, where known:

- September 2018: Possible Double-spend in Bitcoin, Bitcoin ABC and Bitcoin Unlimited
- August 2018: a Brief History of 51% Attacks
- July 2018: KICKICO Hacked ($7.7 Million)
  - gained access to KICK smart-contract account
- July 2018: Bancor Hacked ($13.5 Million)
  - wallet used to upgrade some smart contracts was compromised
- June 2018: Zen Cash 51% Attack ($700,000)
  - reorganized the blockchain by 38 blocks
  - Zen Cash increased required transaction confirmations to 100
- June 2018: Bithumb Hacked ($31.5 Million, second time)
- June 2018: Conrail Hacked ($40 Million)
- June 2018: Shopin Hacked ($10 Million)
  - hot wallet of a major Shopin partner was hacked
- June 2018: Cryptocurrency Mining Malware
- June 2018: Bitfinex DDoS Attacks
- May 2018: Bitcoin Gold 51% Attack ($17.5 Million)
  - reverted 22 blocks
  - Bitcoin Gold advised to increase required transaction confirmations to 50
- May 2018: Monacoin Selfish Mining Attack ($90,000)
- May 2018: Litecoin Cash 51% Attack
  - Litecoin advised to increase required transactions confirmations to 100
- April and May 2018: Verge 51% Attack ($2.85 Million)
- February 2018: Bitgrail Hacked ($195 Million)
  - suspected management fraud

- January 2018: Coincheck Hacked ($530 Million)
  - Security lapse: kept customer assets in hot wallet
- 2017: Anti-Phishing Working Group Report
- December: 2017 NiceHash Hacked ($70 Million)
  - infiltrated system through a compromised company computer using credentials of one of the company's employees
- November 2017: Tether Hacked ($30 Million)
- November 2017: CryptoShuffler Trojan Steals Cryptocurrency ($140,000)
  - clipboard hijacking: replacing the address with its own in the clipboard of the users device
- November 2017: Parity Wallet Bug ($280 Million Frozen)
  - vulnerability in wallet that allowed users to change code and become the owners of wallets that didn't belong to them
  - Interesting quote: Litecoin creator Charlie Lee said that the breach confirmed that the complexity of Solidity, the native programming language of Ethereum, makes the platform a "hacker paradise".
- August 2017: Enigma ICO Hacked ($500,000)
  - attacker compromised Enigma's social accounts, and started to send emails and post Slack messages to urge the community to send funds to his Ethereum address claiming Enigma had opened its pre-ICO
- July 2017: CoinDash ICO Hacked ($7 Million)
  - in the middle of its initial coin offering, hackers managed to change the destination address of ICO funds on its website
- July 2017: Veritaseum ICO Hacked ($8 Million)
  - supplier of the company was compromised
- July 2017: Bithump Hacked (first hack)
  - stole a database of user information off the personal computer of an employee
- July 2017: Parity Multi-Sig Wallet Hack ($32 Million)
- April 2017: Yapizon Hacked ($5 Million)
  - four hot wallets of the exchange were hacked by an unknown group of hackers
- August 2016: Bitfinex Hacked ($72 Million)
  - stolen from users' segregated wallets
- July 2016: Steemit Hacked ($85,000)
- June 2016: DAO Hacked ($50 Million)
  - bug that was identified in the code before the hack, but ignored
- January 2015: Bitstamp Hacked ($5 Million)
- March 2014: Mt. Gox Hack ($473 Million)
  - a flaw in bitcoin itself (transaction malleability), seemingly compounded by MtGox's implementation of the protocol and the company's bizarre internal practices
- August 2010: Bitcoin Hack (184 Billion Bitcoin)
  - On 6 August 2010, a major vulnerability in the bitcoin protocol was spotted. Transactions weren't properly verified before they were included in the transaction log or blockchain, which let users bypass bitcoin's economic restrictions

and create an indefinite number of bitcoins (integer overflow allowed attackers to bypass restrictions). On 15 August, the vulnerability was exploited; over 184 billion bitcoins were generated in a transaction, and sent to two addresses on the network.

Attack details

September 2018: Possible Double-spend in Bitcoin, Bitcoin ABC and Bitcoin Unlimited

https://bitcoincore.org/en/2018/09/20/notice/

> CVE-2018-17144, a fix for which was released on September 18th in Bitcoin Core versions 0.16.3 and 0.17.0rc4, includes both a Denial of Service component and a critical inflation vulnerability.

https://nvd.nist.gov/vuln/detail/CVE-2018-17144

> Bitcoin Core 0.14.x before 0.14.3, 0.15.x before 0.15.2, and 0.16.x before 0.16.3 and Bitcoin Knots 0.14.x through 0.16.x before 0.16.3 allow a remote denial of service (application crash) exploitable by miners via duplicate input. An attacker can make bitcoind or Bitcoin-Qt crash.

August 2018: a Brief History of 51% Attacks

https://www.crypto-news.net/strength-in-numbers-a-brief-history-of-51-attacks/

> Besides CoiledCoin, projects like Terracoin, Feathercoin, and many others have fallen victim to a majority attack. One of the standout examples was executed against the Krypton network, which was subjected to a less common attack that used a new dual-pronged approach, combining majority hashing power with a distributed denial of service (DDoS) to existing nodes to artificially increase the relative hashing power of the attacking party. During this attack, around 21,000 KR was stolen from the Krypton blockchain, which was sent to Bittrex and exchanged for Bitcoin, after which the attackers reversed the transactions by rolling back the blockchain, before making off with the Bitcoin. Following this event, **Krypton suggested that all exchanges raise the minimum confirmation amount to 1000, to increase the difficulty in reverting the blockchain to an earlier state**. Many believe that the Krypton attack was, in fact, a dry-run for a future attack on Ethereum, something that has still yet to occur.

July 2018: KICKICO Hacked ($7.7 Million)

https://www.ccn.com/another-ico-hacked-kickico-loses-8-million-after-smart-contract-breach/

> KICKICO, an initial coin offering (ICO) project launched on top of the Ethereum blockchain protocol, was hacked on July 27, losing more than 70 million KICK worth $7.7 million.

https://medium.com/@kickico/kickico-security-breach-issue-under-control-all-kickcoins-will-be-returned-ebe65a491dec

> On July 26 at 9:04 (UTC) KICKICO has experienced a security breach, which resulted in the attackers gaining access to the account of the KICK smart contract – tokens of the KICKICO platform. The team learned about this incident after the complaints of several victims, who did not find tokens worth 800 thousand dollars in their wallets.
>
> During the investigation, it was found that the total amount of stolen funds is 70,000,000 KICK, which at the current exchange rate is equivalent to $7.7 million.

July 2018: Bancor Hacked ($13.5 Million)

https://www.ccn.com/decentralized-crypto-exchange-bancor-hacked-12m-in-ether-stolen/

> In a tweet earlier today, Bancor stated that it has identified a security breach and will investigate into the issue.
>
> The hack occurred at approximately 00:00 UTC when a wallet used to upgrade some smart contracts was compromised. The wallet has withdrawn ethereum and ERC-20 tokens NPXS and BNT.

https://www.ccn.com/another-ico-hacked-kickico-loses-8-million-after-smart-contract-breach/

> On July 10, just about three and a half weeks ago, Bancor, the fourth largest ICO of all time that raised $150 million, was hacked, losing $13.5 million of its own funds to a group of hackers.
>
> "A wallet used to upgrade some smart contracts was compromised. This compromised wallet was then used to withdraw ETH from the BNT smart contract in the amount of $12.5 million," the official statement of Bancor read.
>
> Fortunately, in the case of Bancor, no user wallets and funds were compromised or stolen, but the situation triggered cryptocurrency researchers and experts

to criticize the structure of the Bancor network.

June 2018: Zen Cash 51% Attack ($700,000)

https://blog.zencash.com/zencash-statement-on-double-spend-attack/?utm_source=blog&utm_medium=social&utm_content=ann&utm_campaign=double+spend

> The Zen network was the target of a 51% attack on 2 June at approximately 8:26 pm EDT (03 June 00:26 hrs UTC). The Zen team immediately executed mitigation procedures to significantly increase the difficulty of future attacks on the network.
>
> A 51% attack or double spend is a major risk for all distributed, public blockchains. All Equihash-based networks are exposed to an influx of new Equihash power and therefore the best short-term mitigation strategy is to recommend that all exchanges **increase their minimum required confirmations to at least 100**.

https://www.ccn.com/zencash-latest-altcoin-to-suffer-51-percent-attack/

> Privacy-centric cryptocurrency ZenCash (ZEN) has become the latest altcoin to succumb to a 51 percent attack, the project's developers confirmed over the weekend.
>
> According to an official statement, a malicious miner successfully executed the attack – as well as at least three double spends – against the cryptocurrency's network on June 2 at approximately 10:43 pm ET.
>
> In this specific case, the attacker executed at least three double spends, including one that **reorganized the blockchain by a full 38 blocks**. Altogether, the attacker made off with more than 23,152 ZEN from these exploits, worth approximately $700,000 at the time of the attack.

June 2018: Bithumb Hacked ($31.5 Million, second time)

https://www.reuters.com/article/us-crypto-currencies-southkorea/hackers-hit-south-korean-cryptocurrency-exchange-bithumb-bitcoin-falls-idUSKBN1JG07F

> SEOUL (Reuters) - South Korean cryptocurrency exchange Bithumb said 35 billion won ($31.5 million) worth of virtual coins were stolen by hackers, the second local exchange targeted in just over a week as cyber thieves exposed the high risks of trading the digital asset.

June 2018: Conrail Hacked ($40 Million)

https://www.reuters.com/article/markets-bitcoin-korea/s-korean-exchange-coinrail-says-hit-by-hackers-bitcoin-slides-idUSL4N1TD066

> SEOUL, June 11 (Reuters) - South Korean cryptocurrency exchange Coinrail said it was hacked over the weekend, prompting an extended sell-off of bitcoin to a 2-month low amid growing concerns about security at small- to mid-sized virtual currency exchanges.
>
> In a statement on its website on Monday, Coinrail said its system was under "cyber intrusion," causing a loss for about 30 percent of the coins traded on the exchange. It did not quantify its value, but said the hack occurred on Sunday.

https://bitcoinmagazine.com/articles/south-korean-exchange-coinrail-hacked-40-million-crypto-reported-stolen/

> South Korean cryptocurrency exchange Coinrail reported a hack on its website during the early morning hours of June 10, 2018. The thieves allegedly made off with over $40 million worth of altcoins and assorted tokens.
>
> Executives announced that roughly 30 percent of the tokens the exchange was housing have been taken, which amounted to nearly $20 million worth of NPXS (Pundi X) tokens, $14 million of Aston X, $6 million in tokens for Dent and over $1 million TRON. At press time, an investigation is underway, and law enforcement officials are working to figure out who was behind the attack.

June 2018: Shopin Hacked ($10 Million)

https://bitcoinmagazine.com/articles/japanese-syndicate-wallet-hacked-10-million-reported-missing/

> Shopin — a universal shopper profile that delivers personal shopping experiences through retailers' apps, websites and stores — says one of its token distributors has been hacked and roughly $10 million in a variety of cryptocurrencies has been stolen.
>
> Representatives of the platform have released the following statement:
>
> "On Wednesday, May 30, 2018, Shopin distributed tokens to one of its leading partners in Japan, who runs a large Japanese syndicate. A few day later, her wallet was hacked, as she was not storing it in cold storage or in a hardware wallet. This is a very sad moment; the Shopin team has a lot of empathy for the situation and the wonderful Japanese people who have participated. We are investigating what can be done to help with the situation."

June 2018: Cryptocurrency Mining Malware

https://www.forbes.com/sites/forbestechcouncil/2018/06/07/a-new-age-of-malware-cryptocurrency-mining/#6f9959571710

> It is no surprise that crypto-malware has been proliferating, as digital currencies provide a level of anonymity and are rather profitable. It is, however, probably the worst of all malware. This new age of crypto-jacking malware simply uses the end user's device to mine cryptocurrency when they visit an infected site.
>
> More websites are adopting cryptocurrency mining through visitors instead of running ads to fund their businesses. Recently, the popular torrent site The Pirate Bay ran a bitcoin-miner as an alternative to ads to generate funds for the business. This new income-generating scheme caused users' central processing units (CPUs) and electricity usages to skyrocket while degrading the performance of their device. Coincidentally, advertising revenue is dropping significantly.

June 2018: Bitfinex DDoS Attacks

https://www.bloomberg.com/news/articles/2018-06-05/crypto-exchange-bitfinex-suspects-it-s-being-attacked-again

> Bitfinex, one of the largest cryptocurrency exchanges, said it has resumed normal operations after suffering a so-called denial-of-service attack earlier.
>
> "The Bitfinex exchange was the target of a Distributed Denial of Service (DDoS) attack this morning," Kasper Rasmussen, head of marketing, said in an e-mailed statement. "The exchange was offline for an hour following the DDoS attack; however, the exchange is back online now. The attack only impacted trading operations, and user accounts and their associated funds/account balances were not at risk at any point during the attack."
>
> The exchange suffered multiple DDoS attacks late last year.

May 2018: Bitcoin Gold 51% Attack ($17.5 Million)

https://forum.bitcoingold.org/t/double-spend-attacks-on-exchanges/1362

> "An unknown party with access to very large amounts of hashpower is trying to use"51 percent attacks" to perform "double spend" attacks to steal money from exchanges," the team says. "We have been advising all exchanges to increase confirmations and carefully review large deposits."

https://forum.bitcoingold.org/t/double-spend-attacks-on-exchanges/1362/21

> The largest observed attack reverted 22 blocks. {-} We advise raising confirmation requirements to 50 at this time, and for special attention to be paid to large transactions.

http://fortune.com/2018/05/29/bitcoin-gold-hack/

> Hackers compromised the cryptocurrency Bitcoin Gold—a lesser known offshoot of the original Bitcoin—this month, using superior computing power to falsify the currency's ledger and swindle at least $18 million from online exchanges.

https://www.bloomberg.com/news/articles/2018-05-29/cryptocurrency-attacks-are-rising-as-rouge-miners-exploit-flaw

> Over the past few weeks, rogue operators of some of the computer networks that perform the complex calculations that verify transactions for various coins are attacking their own networks again. This time it's Bitcoin Gold, an offshoot of the most widely known form of digital money, with a $717 million market capitalization.
>
> Such 51 percent attacks, in which so-called miners gain control of the majority of the network's computing power to falsify transactions, are generating ill-gotten gains that risk collapsing the value of the coins. Under attack for more than a week, Bitcoin Gold is down about 25 percent since May 18.
>
> Similar attacks have targeted Verge, Monacoin and Electroneum, according to Autonomous Research LLC. To gain power over a coin with a market cap of $500 million, an attacker may need to spend as little as $778 an hour, according to Autonomous.

May 2018: Monacoin Selfish Mining Attack ($90,000)

https://www.ccn.com/japanese-cryptocurrency-monacoin-hit-by-selfish-mining-attack/

> Between May 13th and 15th, Monacoin, a cryptocurrency developed in Japan, appears to have suffered from a network attack that caused roughly $90,000 in damages.
>
> The attack appears to have been a selfish mining attack, where one miner successfully mines a block on the blockchain but does not broadcast the new block to other miners. If the secret miner can then find a second block before the rest of the miners find any new blocks, then the secret miner has now effectively created a branch in the chain that is longer than the chain everyone else is working on.
>
> As is standard in most blockchain protocols, the chain with more blocks is

considered by the mining network to be the correct chain, as it has the most "proof of work." So, when the secret miner makes their longer chain public, it invalidates any and all of the blocks discovered by other miners during the time the secret chain was hidden.

It appears the attacker had been trying for half a year to attempt to exploit a weakness in the way Monacoin adjusts its difficulty.

May 2018: Litecoin Cash 51% Attack

https://www.ccn.com/litecoin-cash-latest-small-cap-altcoin-to-suffer-51-percent-attack/

With all the media coverage surrounding double spend attacks in recent weeks, it appears that one such exploit — a 51 percent attack on litecoin cash (LCC) — managed to slip through the cracks.

LCC's developers further said that while a long-term fix will likely require a hard fork, users should wait at least 100 blocks before feeling confident that a transaction has been confirmed and cannot be reversed by a malicious miner.

April and May 2018: Verge 51% Attack ($2.85 Million)

https://www.ccn.com/privacy-coin-verge-succumbs-to-51-attack-again/

Privacy-centric cryptocurrency Verge (XVG) appears to have succumbed to a 51 percent attack for the second time since the beginning of April.

The attack appears to have been carried out between blocks 2155850 and 2206272, enabling the attacker to abscond with approximately 35 million XVG — worth $1.75 million at the current exchange rate — in just a few hours.

The attack appears similar to the one experienced by the Verge network less than two months ago when a malicious miner acquired 20 million XVG, worth more than $1.1 million at the time.

February 2018: Bitgrail Hacked ($195 Million)

http://fortune.com/2018/02/11/bitgrail-cryptocurrency-claims-hack/

An obscure Italian cryptocurrency exchange called BitGrail claims that it was hacked late last week and lost roughly $195 million worth of customers' cryptocurrency. But others — including the developers who created the stolen cryptocurrency — have suggested not all may be as it appears.

Last Thursday, BitGrail founder Francesco Firano claims to have discovered that 17 million Nano tokens, then worth roughly $195 million, had been stolen by hackers.

But that claim has been greeted with widespread skepticism, fueled in part by recent suspicious moves by BitGrail.

According to cryptocurrency news site The Merkle, at least some users at that time were already suspicious that the site was maneuvering towards a so-called "exit scam," and the price of Nano dropped by 20% on the news.

Then, in the wake of the alleged hack, Firano asked the developers of the Nano currency to "fork" their records to restore the funds supposedly stolen from the exchange. This is an eyebrow-raising request, since the immutability of transaction records is one of the core features of cryptocurrency, and held as sacrosanct by many supporters of the technology.

The Nano team responded by publicly rejecting the request, sharing a copy of their communication with Firano, and furthermore alleging that "we now have sufficient reason to believe that Firano has been misleading the Nano Core Team and the community regarding the solvency of the BitGrail exchange for a significant period of time."

January 2018: Coincheck Hacked ($530 Million)

https://www.reuters.com/article/us-japan-cryptocurrency-q-a/the-coincheck-hack-and-the-issue-with-crypto-assets-on-centralized-exchanges-idUSKBN1FI0K4

HONG KONG/TOKYO (Reuters) - Hackers have stolen roughly 58 billion yen ($532.6 million) from Tokyo-based cryptocurrency exchange Coincheck Inc, raising questions about security and regulatory protection in the emerging market of digital assets.

http://fortune.com/2018/01/31/coincheck-hack-how/

Early Friday morning in Tokyo, hackers broke into a cryptocurrency exchange called Coincheck Inc. and made off with nearly $500 million in digital tokens. It's one of the biggest heists in history, with the exchange losing more than 500 million of the somewhat obscure NEM coins. The hack has raised questions about security of cryptocurrencies around the world.

Coincheck hasn't disclosed how their system was breached beyond saying that it wasn't an inside job. The company did own up to a security lapse that allowed the thief to seize such a large sum: It kept customer assets in what's known as a hot wallet, which is connected to external networks. Exchanges generally try to keep a majority of customer deposits in cold wallets, which aren't connected to the outside world and thus are less vulnerable to hacks.

Coincheck also lacked multi-signature security, a measure requiring multiple sign-offs before funds can be moved.

2017: Anti-Phishing Working Group Report

https://www.reuters.com/article/us-crypto-currency-crime/about-1-2-billion-in-cryptocurrency-stolen-since-2017-cybercrime-group-idUSKCN1IP2LU

> NEW YORK (Reuters) - Criminals have stolen about $1.2 billion in cryptocurrencies since the beginning of 2017, as bitcoin's popularity and the emergence of more than 1,500 digital tokens have put the spotlight on the unregulated sector, according to estimates from the Anti-Phishing Working Group released on Thursday.

December 2017: NiceHash Hacked ($70 Million)

https://money.cnn.com/2017/12/07/technology/nicehash-bitcoin-theft-hacking/index.html

> Hackers have carried out a heist on a leading digital currency platform, making off with bitcoins worth more than $70 million.
>
> "Yesterday morning at about 1 a.m. a hacker or a group of hackers was able to infiltrate our systems through a compromised company computer," NiceHash CEO Marko Kobal said in a video statement Thursday.
>
> Roughly 4,700 bitcoins were stolen from the site's account, the CEO said. They're worth roughly $75 million as of Friday afternoon in Asia.
>
> The hackers appear to have entered the NiceHash system using the credentials of one of the company's engineers.

November 2017: Tether Hacked ($30 Million)

https://www.cnbc.com/2017/11/21/tether-hack-attacker-reportedly-steals-30-million-of-digital-tokens.html

> Tether, a start-up that offers dollar-backed digital tokens, claimed Monday that its systems had been hacked, according to cryptocurrency news site CoinDesk.
>
> Tether reportedly claimed $30.95 million worth of its tokens had been stolen.
>
> According to a now deleted post on its website, Tether said a "malicious action by an external attacker" resulted in the theft of the tokens, CoinDesk said.

November 2017: CryptoShuffler Trojan Steals Cryptocurrency ($140,000)

https://www.ccn.com/kaspersky-lab-discovers-new-malware-that-stole-140000-worth-of-bitcoin/

> A new CryptoShuffler Trojan has been discovered that steals cryptocurrency from wallets by replacing the address with its own in the clipboard of the device, reported Kaspersky Lab, which discovered the malware.
>
> Fraudsters using CryptoShuffler Trojan have already stolen 23 BTC, worth around $140,000, from wallets. The creator of the malware has been operating for a year, targeting bitcoin, Ethereum, Dash, Monero, Dash and other cryptocurrencies, according to Kaspersky Lab.
>
> The "clipboard hijacking" technique has been witnessed previously, targeting online payment systems.

November 2017: Parity Wallet Bug ($280 Million Frozen)

https://www.cnbc.com/2017/11/08/accidental-bug-may-have-frozen-280-worth-of-ether-on-parity-wallet.html

> Millions of dollars' worth of ether, the digital token of the ethereum blockchain, could be frozen on a cryptocurrency wallet because one individual "accidentally" triggered a bug.
>
> Parity, a cryptocurrency wallet provider, said in a security alert Tuesday that it had discovered a "vulnerability" in its wallet that allowed users to change code and become the owners of wallets that didn't belong to them.
>
> The company said that one person "suicided" the wallet, deleting its code and freezing all ether tokens contained within. Users are now unable to move funds out of the wallet.

https://www.ccn.com/i-accidentally-killed-it-parity-wallet-bug-locks-150-million-in-ether/

> This is not the first time a bug in Parity's multi-sig wallet code has caused users to lose funds. Earlier this year, an attacker exploited the multi-sig code to steal more than $30 million worth of ether and could have made off with more money if white hat hackers had not drained affected accounts and returned funds to users. At the time, Litecoin creator Charlie Lee said that the breach confirmed that the complexity of Solidity, the native programming language of Ethereum, makes the platform a "hacker paradise".

August 2017: Enigma ICO Hacked ($500,000)

https://www.ccn.com/hacker-nets-over-500000-after-hacking-enigma-before-its-ico-date/

> Last month, CCN reported on CoinDash's ICO being hacked. Hackers managed to change the address on its website and made over $9 million. Now, despite not making as much money, a hacker managed to compromise Enigma before its ICO in a similar way, and has netted over 1,500 Ether (over $500,000) from the community.
>
> After the hacker managed to compromise Enigma's social accounts, he started to send emails and post Slack messages to urge the community to send funds to his Ethereum address claiming Enigma opened its pre-ICO. In sent emails, according to reports, the hacker stated it had a hard cap set at $20 million.

July 2017: CoinDash ICO Hacked ($7 Million)

http://fortune.com/2017/07/18/ethereum-coindash-ico-hack/

> Hackers hijacked cryptocurrency trading platform CoinDash on Monday just as it was in the middle of its initial coin offering, or ICO. It's the first known breach of an ICO, this season's hottest fundraising method.
>
> CoinDash, an Israeli startup, planned to raise capital by selling its own digital tokens in exchange for the cryptocurrency Ethereum, which is similar to Bitcoin. But just 13 minutes into the token sale, which began at 9 a.m. ET Monday, an "unknown perpetrator" hacked CoinDash's website and changed the address for sending investments to a fake one, the company later announced on its website. That diverted millions of dollars in contributions to the attacker.
>
> While the CoinDash ICO still managed to raise $6.4 million from early investors, the hacker stole $7 million worth of Ethereum before the company was forced to pull the plug on the token sale. Despite the losses, CoinDash promised to dole out its tokens accordingly to everyone who participated in the ICO before it was shut down, whether or not they sent funds to the correct address.

July 2017: Veritaseum ICO Hacked ($8 Million)

https://www.coindesk.com/veritaseum-founder-claims-8-million-ico-token-stolen/

> Yet another initial coin offering (ICO) project is claiming that it has been the victim of a hack.

Just a week after a prominent ICO saw its sale disrupted, the team behind Veritaseum, the issuer of a cryptocurrency called VERI, is claiming 36,000 of its tokens were stolen and subsequently exchanged for ether.

After one user posted a screen capture of Veritaseum founder Reggie Middleton mentioning the hack on Slack, Middleton later confirmed the theft. There, he said the hackers "dumped" the tokens within a few hours and without the public knowing about the hack.

Based on the address of the hacker or hackers that Middleton located in his post, all the stolen tokens were exchanged on EtherDelta, a decentralized trading platform, of which 80 percent of the volume is in VERI. The exchange only supports the VERI to ETH trading pair.

He continued to say that a supplier of the company was compromised, and that this led to the hack, but refused to offer additional details such as the name the company or the attack vector.

July 2017: Bithump Hacked (first hack)

http://fortune.com/2017/07/05/bitcoin-ethereum-bithumb-hack/

Hackers have stolen user data and money from Bithumb, one of the top five biggest Ethereum and Bitcoin cryptocurrency exchanges, the company said Friday.

Bithumb said that the thieves stole a database of user information off the personal computer of an employee, rather than off the company's internal network. The attackers allegedly nabbed the names, email addresses, and mobile phone numbers (no passwords, apparently), of more than 31,800 customers, according to a Monday report from Yonhap, a South Korean news wire.

July 2017: Parity Multi-Sig Wallet Hack ($32 Million)

https://www.ccn.com/hackers-seize-32-million-in-parity-wallet-breach/

Parity Technologies has issued a critical security alert for their popular multi-signature wallet software following an apparent breach. The development team urged users holding ether in multi-sig contract wallets version 1.5 or later to immediately transfer their tokens to a secure address. The breach only affects multi-sig wallets; normal wallets appear to be safe.

According to Etherscan.io, the hacker(s) absconded with more than 153,000 ether, worth more than $32 million at the current exchange rate ($213 per CoinMarketCap).

> Non-ether tokens held in multi-sig wallets are also subject to the vulnerability, but the hacker only stole a small amount of one [other] token.

April 2017: Yapizon Hacked ($5 Million)

https://www.ccn.com/south-korea-yapizon-bitcoin-exchange-hack/

> According to the exchange's official statement translated by CCN, four hot wallets of the exchange were hacked by an unknown group of hackers. The Yapizon legal team noted that approximately 37 percent of user funds were hacked and exactly 3,816 bitcoins were stolen from the exchange.

August 2016: Bitfinex Hacked ($72 Million)

http://fortune.com/2016/08/03/bitcoin-stolen-bitfinex-hack-hong-kong/

> Nearly 120,000 units of digital currency bitcoin worth about US$72 million was stolen from the exchange platform Bitfinex in Hong Kong, rattling the global bitcoin community in the second-biggest security breach ever of such an exchange.
>
> Zane Tackett, Director of Community & Product Development for Bitfinex, told Reuters on Wednesday that 119,756 bitcoin had been stolen from users' accounts and that the exchange had not yet decided how to address customer losses.
>
> "The bitcoin was stolen from users' segregated wallets," he said.

July 2016: Steemit Hacked ($85,000)

https://www.ccn.com/steemit-secures-hacked-accounts-advises-new-password/

> Steemit, the blockchain-based social media platform that was targeted in a cyberattack saw malicious hackers hack user accounts and their balances. Since then, a recovery process has seen compromised accounts with balances over $100 USD, restored and refunded to the affected users.
>
> Blockchain-powered Steemit saw some 250 user accounts compromised along with $85,000 in Steem dollars and the token cryptocurrency Steem, stolen, in a cyber attack yesterday.

June 2016: DAO Hacked ($50 Million)

https://www.nytimes.com/2016/06/18/business/dealbook/hacker-may-have-removed-more-than-50-million-from-experimental-cybercurrency-project.html

> A hacker on Friday siphoned more than $50 million of digital money away from an experimental virtual currency project that had been billed as the most successful crowdfunding venture ever — taking with him not just a third of the venture's money but also the hopes and dreams of thousands of participants who wanted to prove the safety and security of digital currency.
>
> The attack most likely puts an end to the project, known as the Decentralized Autonomous Organization, which had raised $160 million in the form of Ether, an alternative to the digital currency Bitcoin. While the computer scientists involved in the project are aiming to tweak the code that underpins Ether in a way that will recover the money, the theft is nevertheless prompting a bigger debate about the viability and principles of virtual currencies like Bitcoin and Ether.
>
> "This is one of the nightmare scenarios everyone was worried about: Someone exploited a weakness in the code of the D.A.O. to empty out a large sum," Emin Gün Sirer, a computer science professor at Cornell who co-wrote a paper pointing out problems with the project, said on Friday.

January 2015: Bitstamp Hacked ($5 Million)

http://fortune.com/2015/01/05/bitstamp-bitcoin-freeze-hack/

> Hackers have stolen more than $5 million in virtual currency from Bitstamp, a major bitcoin exchange, forcing the company to freeze user accounts, suspend trades and block deposits.
>
> The Slovenia-based company said Monday that fraudsters made off with 19,000 bitcoins a day prior. It was not immediately clear who was responsible for the theft or how it happened.

March 2014: Mt. Gox Hack ($473 Million)

https://www.theguardian.com/technology/2014/feb/27/how-does-a-bug-in-bitcoin-lead-to-mtgoxs-collapse

> The story behind the collapse of MtGox is almost unbelievable. How could any business not notice that over £200m worth of assets had simply disappeared?
>
> The answer involves a flaw in bitcoin itself, seemingly compounded by MtGox's implementation of the protocol and the company's bizarre internal practices.

Between them, they created a situation where a cunning attacker could convince the company to hand over money without even realising what it was doing.

At the heart of Mt Gox's troubles lies an issue with bitcoin known as "transaction malleability".

August 2010: Bitcoin Hack (184 Billion Bitcoin)

https://en.wikipedia.org/wiki/History_of_bitcoin

On 6 August 2010, a major vulnerability in the bitcoin protocol was spotted. Transactions weren't properly verified before they were included in the transaction log or blockchain, which let users bypass bitcoin's economic restrictions and create an indefinite number of bitcoins. On 15 August, the vulnerability was exploited; over 184 billion bitcoins were generated in a transaction, and sent to two addresses on the network. Within hours, the transaction was spotted and erased from the transaction log after the bug was fixed and the network forked to an updated version of the bitcoin protocol.

https://nvd.nist.gov/vuln/detail/CVE-2010-5139

Integer overflow in wxBitcoin and bitcoind before 0.3.11 allows remote attackers to bypass intended economic restrictions and create many bitcoins via a crafted Bitcoin transaction.

References

[Ber+12] Daniel J Bernstein et al. "High-speed high-security signatures". In: *Journal of Cryptographic Engineering* 2.2 (2012), pp. 77–89. URL: https://ed25519.cr.yp.to/ed25519-20110926.pdf.

[Bra87] Gabriel Bracha. "Asynchronous Byzantine Agreement Protocols". In: *Inf. Comput.* 75.2 (1987), pp. 130–143. DOI: 10.1016/0890-5401(87)90054-X. URL: https://doi.org/10.1016/0890-5401(87)90054-X.

[But13] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. 2013. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[Cra+18] Tyler Crain et al. *DBFT: Efficient Byzantine Consensus with a Weak Coordinator and its Application to Consortium Blockchains*. 2018. URL: https://arxiv.org/pdf/1702.03068v3.pdf.

[Gon08] Georges Gonthier. "Formal Proof — The Four-Color Theorem". In: 2008. URL: https://www.ams.org/notices/200811/tx081101382p.pdf.

[JL17] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. RFC Editor, Jan. 2017.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (1982), pp. 382–401. DOI: 10.1145/357172.357176. URL: http://doi.acm.org/10.1145/357172.357176.

[Nak08] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[Qu99] Minghua Qu. "SEC 2: Recommended elliptic curve domain parameters". In: *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6* (1999).

[Spe+17] Antal Spector-Zabusky et al. "Total Haskell is Reasonable Coq". In: *CoRR* abs/1711.09286 (2017). arXiv: 1711.09286. URL: http://arxiv.org/abs/1711.09286.