Zold: A Fast Cryptocurrency for Micro Payments

Yegor Bugayenko
yegor256@gmail.com

0.2.5

June 27, 2018

Abstract

In the last few years digital currencies have successfully demonstrated their ability to become an alternative financial instrument in many different markets. Most of the technologies available at the moment are based on the principles of Blockchain architecture, including dominating currencies like Bitcoin and Etherium. Despite its popularity, Blockchain is not the best possible solution for all scenarios. One such example is for fast micro-payments. Zold is an *experimental* alternative that enables distributed transactions between anonymous users, making micro-payments financially feasible and fast. It borrows the "proof of work" principle from Bitcoin, and suggests a different architecture for digital wallet maintenance.

1 Motivation

Bitcoin, the first decentralized digital currency, was released in January 2009 (Nakamoto 2008). In the following years "a libertarian fairy tale" and "a simple Silicon Valley exercise in hype" turned into "a catalyst to reshape the financial system in ways that are more powerful for individuals and businesses alike," according to Andreessen (2014). Even though Cheah et al.

(2015) argues that "the fundamental value of Bitcoin is zero" , it seems that "the question is not whether Bitcoin has value; it already does. The question is whether the efficiencies of a cybercurrency like Bitcoin can be merged with the certainties of an honest central bank." (Van Alstyne 2014).

The core component of Bitcoin is Blockchain technology, which "ensures the elimination of the double-spend problem, with the help of public-key cryptography" and "coins are transferred by the digital signature of a hash" (Pilkington 2016). Very soon after Bitcoin was created, similar products were introduced, which were also based on the principles of Blockchain, such as Etherium (Buterin 2013).

Even though Blockchain is a sound solution to the double-spending problem, there could be other solutions, including different "proof-of" alternatives.[1] For example, Everaere et al. (2010) gave a summary of them and introduced their own; Boyen et al. (2016) described "a truly distributed ledger system based on a lean graph of cross-verifying transactions"; recently IOTA, a "tangle-based cryptocurrency," was launched (Popov 2017).

Zold is also a decentralized digital currency that maintains its ledgers through an unpredicable amount of anonymous and untrustable server nodes, trying to guarantee data consistency. The architecture of Zold is not Blockchain-based. The development of Zold was motivated by the desire to overcome two obvious disadvantages present in the majority of all existing cryptocyrrencies:

The first problem is that transaction processing is rather slow.[2] Karame et al. (2012) says that "Bitcoin requires tens of minutes to verify a transaction and is therefore inappropriate for fast payments." It is inevitable, since "processing speed is at odds with the security aspects of the underlying proof-of-work based consensus mechanism" according to Kiayias et al. (2015). Ethereum, according to Fekkes et al. (2018), can process "two times more transactions per second than Bitcoin is able to do," but this still is rather

[1] `https://goo.gl/aqzf2Q`: "Proof-of-Burn": instead of bringing the money together into computer equipment, the owner burns the coins by sending to an address where they are irretrievable. By doing this, the owner gets a privilege to mine on the system. "Proof-of-Stake": the coins exist from the start, and the validators get a reward in the form of transaction fees. "Proof-Of-Capacity": one pays with the hard drive space. The more dedicated hard drive space, the higher probability of mining the next block and earning a reward. "Proof-of-Elapsed-Time": one uses a Trusted Execution Environment or TEE to ensure a random looter production.

[2] `https://goo.gl/sWiAWc`: "Current rates for Bitcoin processing speed is 7 transactions per second (tps) while Paypal handles an average of 115 tps and the VISA network has a peak capacity of 47,000 tps (though it currently needs 2000-4000 tps)."

slow.

The second problem, as noted by Popov (2017), is that "it is not easy to get rid of fees in the blockchain infrastructure since they serve as an incentive for the creators of blocks." As per Möser et al. (2015), "Bitcoin users are encouraged to pay fees to miners, up to 10 cents (of USD) per transaction, irrespective of the amount paid" which especially hurts when transaction amounts are smaller than a dollar. Moreover, according to Kaskaloglu (2014), "an increase in transaction fees of Bitcoin is inevitable."

Thus, the speed is low and the processing fees are high. Zold was created as an attempt to resolve these two problems.

2 Principles

No General Ledger. Unlike *all* other crypto currencies, there is no central ledger in Zold. Each wallet has its own personal ledger. All transactions in each ledger are confirmed by RSA signatures of their owners. Section 4 explains how this works.

Proof of Work. Similar to many other digital currencies—including Bitcoin, Etherium, Monero, Trinity, Plancoin, Dero, and many others—Zold nodes find consensus by using the CPU power invested by each of them to perform certain expensive and meaningless calculations that result in finding hash suffixes. Section 3 describes the algorithm being used.

Detached Operations. To make a payment a user *fetches* a wallet from the network, makes as many payments from the wallet to other wallets as necessary, and *pushes* the wallet back to the network. Thanks to that, there are no technical limitations to the amount of transactions the network can process per second. Section 9 explains the details.

Root Wallet. Zold is a pre-mined[3] digital asset, similar to Ripple,[4] Cardano,

[3] `https://goo.gl/QBhbcT`: "A premine or instamine is where the developer or developers don't release the crypto currency in what can be considered a fair manner. Even Bitcoin can be considered to be instamined to a certain extent. A premine is where a developer allocates a certain amount of currency credit to a particular address before releasing the source code to the open community."

[4] `https://goo.gl/XAtPH8`: "When the Ripple network was created, 100 billion XRP was created. The founders gave 80 billion XRP to the Ripple Labs. Ripple Labs will develop the Ripple software, promote the Ripple payment system, give away XRP, and sell XRP."

Stellar,[5] EOS, NEO, Loki,[6] and many others. The only way to get ZLD is to receive it from someone else. The root wallet belongs to the issuer and may have a negative balance, which can grow according to a pre-defined restrictive formula, explained in Section 5. All other wallets can only have positive balances.

Taxes. Unlike many other payment systems, Zold doesn't require its users to pay transaction fees. Instead, wallets have to pay regular "taxes" for their maintenance. The amount of Taxes depends two factors: 1) the number of transactions in a particular wallet and 2) the age of the wallet. Section 7 provides more details.

No Trust. The network of communicating nodes maintain the wallets of Zold users. Anyone can add a node to the network. It is assumed that any node may contain corrupted data, either by mistake or intentionally. Section 8 explains how nodes communicate and rate each other.

Open Source. Zold is a command line tool. Its entire code base is open source and hosted on its GitHub yegor256/zold repository. There is also a primitive web wallet interface, with GitHub OAuth login: wts.zold.io.

Capacity. One currency unit is called ZLD. One ZLD by convention equals to 2^{32} *zents* (4,294,967,296). All amounts are stored as signed 64-bit integers. Therefore, the technical capacity of the currency is 2,147,483,648 ZLD (two billion).[7]

Hexspeak. A user's Wallet ID is a 16-digit hexadecimal number, which is not restricted by any formula. A user may make up any number, even using Hexspeak.

[5] `https://goo.gl/CnQQwA`: "The stellar network started with 100 billion lumens. There is a 1% p.a. inflation, hence the current total of roughly 103.5 billion lumens. About 18 billion lumens are on the market and the other 85 is held by the stellar development foundation."

[6] `https://goo.gl/By5CR3`: "Over 7 million Loki is held in escrow for the Founder, Advisor, and Seed allocations. The Founder and Advisor allocations follow a 12 month lockup schedule, where 25% of each allocation is released every 90 days following mainnet launch. The allocations to Founders and Advisors are remuneration for services rendered to the LAG Foundation Ltd. The Seed allocation follows a similar schedule, with a 30% initial release and 20% every 90 days until the final release of 10%."

[7] To compare, the total supply of some crypto currencies is: Bitcoin: 21m BTC, Ethereum;: 100m ETH, Ripple: 100b XRP, Litecoin: 84m LTC, Cardano: 31b ADA, Stellar: 103b XLM, NEO: 100m NEO, Dash: 19m DASH.

3 Proof of Work

The system consists of nodes (server machines), which maintain the data. In order to guarantee data consistency among all distributed nodes there has to be an algorithm of data segregation. Corrupted data must be detected earlier and filtered out as quickly as possible. Bitcoin employed this algorithm, which was originally introduced and labeled as *proof of work* by Back (1997).

Its fundamental principle is that each block of data must have a special number attached to it, known as *nonce*, which is rather difficult to calculate, because it requires a lot of CPU power. It is assumed that at any moment of time the majority of nodes in the network invest their CPU power into calculating the nonces for uncorrupted data. If and when for any reason certain data does get corrupted, the amount of CPU power that the corrupted part of the network decides to invest into its nonces calculation would be smaller than what the other part of the network invests into correct data. The latter part will quickly dominate the former and the nodes with corrupted data will be ostracized and eventually ignored (Nakamoto 2008).

Zold has borrowed elements of this principle, but has also modified it. Zold also requires its nodes to invest their CPU power into meaningless and repetative calculations in order to identify which part of the network they belong to: corrupted or not. Each Zold node has to calculate its *score*, which is indicative of the CPU power the node has invested into its calculation.

Similar to Bitcoin nonces, Zold nodes repeatedly calculate cryptographic hashes, looking for consecutive zeros inside them. First, in order to calculate a score, a node makes the *prefix*, which consists of four parts, separated by spaces:

1. The current timestamp in UTC, in ISO 8601,
2. The host name or IP address, e.g. `b2.zold.io`,
3. The TCP port number,
4. The invoice.

For example, the prefix may look like this:

```
2018-06-27T06:22:41Z b2.zold.io 4096 THdonv1E@abcdabcdabcdabcd
```

Then, the node attempts to append any arbitrary text, which has to match `/[a-zA-Z0-9]+/` regular expression, to the end of the prefix and calculates SHA-256 hash of the text in the hexadecimal format. For example, this would be the prefix with the attached `3a934b` suffix:

```
2018-06-27T06:22:41Z b2.zold.io 4096 THdonv1E@abcdabcdabcdabcd
 ↪  3a934b
```

The hash of this text will be (pay attention to the trailing zeroes)[8]:

```
c9c72efbf6beeea13408c5e720ec42aec017c11c3db335e05595c03755000000
```

The node attempts to try different sufficies until one of them produces a hash that ends with a few trailing zeroes. The one above ends with six zeroes.

When the first suffix is found, the score is 1. Then, to increase the score by one, the next suffix has to be found, which can be added to the previous hash in order to obtain a new hash with trailing zeros. For example, adding `...`:

```
c9c72efbf6beeea13408c5e720ec42aec017c11c3db335e05595c03755000000
 ↪  1421217
```

This new SHA-256 input produces the following input, which also ends with six zeroes:

```
e04ab4e69f86aa17be1316a52148e7bc3187c6d3df581d885a862d8850000000
```

And so on.

The score is only valid when the starting time is earlier than the current time, but not earlier than 24 hours ago. The *strength* of the score is the amount of trailing zeros in the hash. In the example above the strength is six. The larger the strength, the more CPU power it takes to earn the score. All nodes in the network must have the same strength of their scores.

[8] You can validate it at this online SHA-256 hash generator: https://goo.gl/QtHd9a

Similar to Bitcoin network, which, according to Hayes (2017), "automatically adjusts the difficulty variable so that one block of bitcoins is found, on average, every ten minutes," Zold network may use the strength in order to calibrate itself.

4 Wallets

There is no central ledger in Zold, unlike many other digital currencies. Instead, users have their own *wallets* (any number of them) with their own ledgers inside. Each wallet is an ASCII-text file with the name equal to the wallet ID. For example, the wallet in the file `12345678abcdef` may include the following text:

```
zold
0.6.4
12345678abcdef
AAAAB3NzaC1yc2EAAAADAQABAAABAQCuLuVr4Tl2sXoN5Zb7b6SKMPrVjLxb...

003a;2017-07-19T21:24:51Z; ffffffff9c0ccccd; Ui0wpLu7;
 ↪  98bb82c81735c4ee; For services;SKMPrVj...
003b;2017-07-19T21:25:07Z; fffffffffa72367; xksQuJa9;
 ↪  98bb82c81735c4ee; For food;QCuLuVr4...
0f34;2017-07-19T21:29:11Z; 0000000000647388; kkIZo09s;
 ↪  18bb82dd1735b6e9; -;
003c;2017-07-19T22:18:43Z; fffffffffff884733; pplIe28s;
 ↪  38ab8fc8e735c4fc; For programming;2sXoN5...
```

Lines are separated by either CR or CRLF. There is a header and a ledger, separated by an empty line. The header includes four lines:

1. Network name, `[a-z]{4,16}`;
2. Software version, `[0-9]+(.[0-9]+){1,2}` (semantic versioning);
3. Wallet ID, a 64-bit unsigned integer in hexadecimal format;
4. Public RSA key of the wallet owner, in Base64.

The ledger includes transactions, one per line. Each transaction line contains fields separated by a semi-colon:

1. `id`: Transaction ID, an unsigned 16-bit integer, 4-symbols hex;
2. `time`: date and time, in ISO 8601 format, 20 symbols;
3. `amount`: Zents, a signed 64-bit integer, 16-symbols hex;
4. `prefix`: Payment prefix, 8-32 symbols;
5. `bnf`: Wallet ID of the beneficiary, 16-symbols hex;
6. `details`: Arbitrary text, matching `/[a-zA-Z0-9 -.]{1,512}/`;
7. `signature`: RSA signature, 684 symbols in Base64.

Transactions with positive amount don't have signatures. Their IDs point to ID fields of corresponding beneficiaries' wallets.

The `prefix` is a piece of text randomly selected from the RSA key of the beneficiary wallet. It is used for security reasons, in order to make impossible "wallet masquerading" (pushing a new wallet with the same ID, but a different key).

The combination `id` + `bnf` must be unique throughout each wallet.

The RSA signature is calculated using the private key of the wallet and the following fields of transaction, separated by spaces: `bnf`, `id`, `time`, `amount`, `prefix`, `bnf`, `details`.

For example, this text may be used as a signing input:

```
12345678abcdef 003a 2017-07-19T21:24:51Z ffffffff9c0ccccd
↪  Ui0wpLu7 98bb82c81735c4ee For services
```

Each transaction takes 1284 symbols at most.

The order of transactions is not important, as long as the final balance is positive.

5 Mining Formula

The only way to get ZLD is to receive it from the *root* wallet with a system predefined ID `0000000000000000`. This is the only wallet on the Zold network

that can carry a negative balance. However, to prevent an uncontrolled emission of ZLD, the balance of this wallet must satisfy the following formula:

$$t = \frac{h}{24 \times 1024}; \quad z = 2^{63} \times (1 - 2^{-t}).$$

Here h is the age of the root wallet in hours and z is the maximum amount of zents it can issue at that moment. The first six years will look like this:

Year	ZLD	Share
1st	470m	22%
2nd	837m	39%
3rd	1.1b	52%
4th	1.3b	63%
5th	1.5b	71%
6th	1.7b	77%

The limitation is hardwired in Zold software and can't be eliminated.

6 Total Supply

Some cryptocurrencies, in order "to protect against inflationary forces," algorithmically limit their total supply (Bohr et al. 2014). For example, Bitcoin halves the creation rate every 210,000 blocks, which started from 50 BTC per block in 2009, making it technically possible to "mine" the total amount of 21 million until 2140 (Iwamura et al. 2014).

Ethereum, to the contrary, is infinite, although its total annual supply is technically limited to 18 million ether. It is suggested by Ethereum creators that potential token losses, caused by loss of keys and codes, death of holders or misuse, "will be compensated for by not having a limit on the total amount of ether that will be created." (Fekkes et al. 2018).

In Zold, as explained in Section 4, any particular transaction can transfer up to 2^{63} zents, which is, by convention, is 2^{32} ZLD. Thus, technically, a summary of transactions in a wallet may be larger than 2^{63}. However, the software will complain and reject a wallet if such an overflow happens. The minimum balance of a wallet is -2^{63} and the maximum is 2^{63}. Since there is only one wallet in the entire network that can legally have a negative balance, the entire "supply" of Zold currency is roughly 2.13 billion ZLD.

7 Taxes

Each wallet must pay *taxes* in order to be promoted by nodes. The maximum amount of tax debt a node can tolerate is 1 ZLD. This means that if the debt is smaller, all nodes must promote the wallet to their remote nodes. If the debt is bigger, a node will reject the wallet, which will make it impossible to make any new payments from it.

The amount of taxes to be paid is calculated by the following formula:

$$X = A \times F \times T.$$

A is the total age of the wallet, which is calculated as the difference in hours between the current time and the time of the oldest transaction in the wallet. T is the total number of transactions in the wallet. F is the fee per transaction/hour, which is equal to 7.48 zents (a one-year-old wallet with 4096 transactions must pay approximately 16 ZLD taxes annually).

In order to pay taxes the owner of the wallet must select a remote node from the network with a score of 16 or more. Then, it has to take the invoice from the score, request the node to lock the score for a minute, and send the payment of 1 ZLD or less to that node. The score with exactly 16 suffixes has to be placed into the details of the transaction, prefixed by `TAXES `.

The most active remote node will be selected as tax receiver. It's up to the payer which node to select. This situation is similar to Bitcoin, where "a miner's chance of winning the competition is (roughly, and with some caveats) equal to the proportion of the total computing power that they control," according to Foroglou et al. (2015).

All tax payments inside a wallet must have unique scores. Duplicate tax payments are ignored.

8 Remote Nodes

Each node maintains a list of *remote nodes* (their host names and TCP port numbers), their scores and their availability information. When the node is first installed, the list contains a limited amount of pre-defined addresses[9]. The list is updated by manual user request and automatically in order to

[9] https://github.com/zold-io/zold/blob/master/resources/remotes

give priority to high-score nodes and the nodes with the highest availability. Moreover, the node adds new elements to the list by retrieving them from all other available remote nodes.

The built-in mechanism focuses on the following factors of remote node *quality* (in order of importance):

1. Visibility: the payer has to know the node;
2. Availability: the amount of errors seen recently;
3. Knowledgeability: the amount of nodes this node is aware of;
4. Activity: the frequency of push requests the node originates;
5. Score: the one reported during the most recent handshake.

Every time a node receives an HTTP request from anyone, it reads the 'X-Zold-Score' HTTP header. The header, if it exists, may include the text representation of a score of another node, which is making the request. The request receiver validates the score and, if its value is bigger than three, adds node coordinates (which are available in the score) to the local list of remote nodes. Thanks to this auto-discovery mechanism nodes become aware of each others presence in the network.

Each node runs a "reconnect" procedure every minute, updating the list of remote nodes and removing those, which have too low availability values.

9 Fetch, Merge, and Propagate

In order to see a wallet, it has to be *fetched* from a number of remote nodes. The nodes may provide different versions of the same wallet, either because certain data is corrupted or because modifications were made to the same wallet from different parts of the network. Each version retrieved from the network is stored in a local *copy* and gets a score assigned to it. The score of the local copy is a summary of all scores of the nodes that provided that copy. Let's say, there are 17 nodes in the network and they provided three different copies of the wallet:

```
copy-1: 78,090 bytes, 11 servers, 177 score
copy-2: 56,113 bytes, 4 servers, 69 score
copy-3: 97,132 bytes, 2 servers, 37 score
```

The fetch operation ends at this point. The next step is to *merge* all three copies into the local one, if it exists. The algorithm of merging is as follows:

First, the copy of the wallet into which we are merging is added to the list, with the score of zero:

```
copy-0: 55,991 bytes, 0 servers, 0 score
copy-1: 78,090 bytes, 11 servers, 177 score
copy-2: 56,113 bytes, 4 servers, 69 score
copy-3: 97,132 bytes, 2 servers, 37 score
```

Then, the copy with the highest score is assumed to be the correct one, which is the `copy-1` in this example.

Then, all other copies, in the order of their scores, are merged into the correct one, transaction by transaction, and the following rules are applied:

1. If the transaction already exists, it's ignored;
2. If the transaction is negative (spending money) and its ID is lower than the maximum ID in the ledger, it gets ignored as a fraudulent one ("double spending");
3. If the transaction makes the balance of the wallet negative, it is ignored;
4. If the transaction is negative and its signature is not valid, it is ignored;
5. If the transaction is positive and it's absent in the paying wallet (which exists at the node), it's ignored; If the paying wallet doesn't exist at the node, the transaction is ignored;
6. Otherwise, it gets added to the end of the ledger.

When the merge process is complete, the modifications get *propagated* to other wallets available locally. Each transaction that has a negative amount is copied to the ledger of their receiving wallets (with a reversed sign) if it doesn't yet exist there.

10 Invoice, Pay, and Push

To send money from one wallet to another, the owner of the sending wallet has to add a negative transaction to it and sign it with the private RSA key.

Each transaction has to have a *payment prefix* attached to it. The prefix is a text block of 8-32 symbols randomly selected from the text representation of the public key of the receiving wallet. This prefix becomes a field in a transaction and participates in the body for the RSA signature. Without this prefix it would be possible to steal a wallet by replacing it with a new one, with a different pair of keys.

An *invoice* is a combination of payment prefix and wallet ID separated by the `@` sign, for example:

```
THdonv1E@abcdabcdabcdabcd
```

Here, `THdonv1E` is the payment prefix taken from the public key of the wallet, and `abcdabcdabcdabcd` is the wallet ID. Obviously, an invoice is valid only if the prefix can be found in the public key of the wallet.

At any moment of time any node may decide to push a wallet to another node. The receiving node accepts it, merges with the local version, and keeps it locally. Then, it *promotes* the wallet to all known remote nodes.

11 RESTful API

There is a limited set of RESTful API entry points in each node. Each response has `Content-Type`, `Content-Length`, and `X-Zold-Version` HTTP headers.

`GET /` is a home page of a node that returns JSON/200 response with the information about the node. For example (other details may be added in further versions):

```
{
  "version": "0.6.1",
  "score": {
    "value": 3,
    "host": "b2.zold.io",
```

```
    "port": 4096,
    "invoice": "THdonv1E@0000000000000000",
    "suffixes": [ "4f9c38", "49c074", "24829a" ],
    "strength": 6,
    "time": "2018-06-20T05:22:54Z"
  }
}
```

`GET /remotes` returns the list of remote nodes known by the node, in JSON/200:

```
{
  "version": "0.6.1",
  "all": [
    { "host": "b2.zold.io", "port": 4096 },
    { "host": "b1.zold.io", "port": 80 }
  ]
```

`GET /wallet/<ID>` returns the content of the wallet, in JSON/200:

```
{
  "version": "0.6.1",
  "body": "..."
```

The `body` includes the entire content of the wallet file, according to the format explained in Section 4.

If the wallet is not found, a 404 HTTP response is returned.

If the client provided the pre-calculated MD5 hash of the wallet content in the `If-None-Match` HTTP header and it matches with the hash of the content the node contains, a 304 HTTP response is returned.

`PUT /wallet/<ID>` pushes the content of the wallet to the node. The node responds either with 202 (if accepted), 400 (if the data is corrupted), 402 (if taxes are not paid), or 304 (if the content is the same as the one the node already has).

12 Incentives

Anonymous users will only participate in Zold and maintain their nodes if they have enough financial motivation to do so. Simply put, their expenses must be lower than the income they are getting in the form of taxes. This Section analyzes the most obvious questions users may have, regarding their motivation.

12.1 To Stay Online

What is the reason for a node to stay online and spend its CPU power and network traffic? Each node is motivated by the hopes that wallet owners will pay taxes to its invoices. The software automatically decides to which node to pay taxes and the selection is made by the availability criteria. The node which is most available and visible will get the majority of tax payments.

Whether the tax payments in ZLD will become a strong enough incentive to support the network is a separate problem, which exists in all other decentralized cryptocurrencies. Iwamura et al. (2014) analyzes this problem and concludes that "if the Bitcoin price drops below a threshold, the Bitcoin system as a whole may collapse." In other words, the owners of nodes will be interested in keeping them online for as long as the market price of ZLD taxes they collect is high enough to cover their hardware and network traffic expenses.

12.2 To Accept Wallets

Why would a node accept push requests and spend its storage space on the wallets coming in? If the node doesn't accept a push request, its availability rating decreases and other nodes will stop paying taxes to it.

12.3 To Advertise Other Nodes

What is the incentive to advertise other remote nodes via the `/remotes` RESTful entry point and why can't a node always return an empty list, expecting its clients to always pay taxes to it? The software automatically prioritizes remote nodes by the amount of other remote nodes it promotes. The longer the list a node returns, the higher its chance to be at the top of

the list.

12.4 To Promote Wallets

What is the incentive to promote wallets to remote nodes, spending network traffic for this operation? Each node also ranks its remote nodes by the amount of push requests they send. Thus, in order to stay on top of these rankings each node is interested in pushing wallets further.

13 Threats and Responses

It is obvious that a distributed system that consists of anonymous nodes even theoretically can't be 100% safe, reliable, secure and trustworthy. Zold is not an exception. However, it's designed in an honest attempt to mitigate all critical threats and make the system "reliable enough." This Section summarizes the most import of those threads and explains how Zold responds to them.

13.1 Double Spending Attack

It is possible to submit the same spending transaction to the same wallet and then push it to two different nodes in different parts of the network. They won't know about each other and will propagate those spending transactions to other wallets. Both two owners of those money receiving wallets will think that their money arrived, while only one of them is a legit receiver, the other transaction is fraudulent.

This will happen, but very soon one part of the network will dominate the other one, and one of the transactions will be rejected from the wallet, after a number of merge operations in all nodes of the network. The receiver of the money must be careful and always do the full fetch (from as many nodes as possible) in order to guarantee safety of transactions.

13.2 51% Attack

A group of nodes can combine their CPU power in order to win the consensus algorithm and add fraudulent incoming transactions to a wallet. The fetching

node will trust the wallet "as is" and will think that the balance of the wallet is larger than it actually is.

This may happen, but a fetching node may always re-validate the entire wallet, by checking RSA signatures of all transactions. This will take some time, but will provide an extra guarantee to the client.

13.3 Fraudulent Tax Refunds

Some nodes may resell their scores to their affiliated tax payers, and they refund them some amount of taxes back. This will be profitable both for the tax payers, since they will pay less taxes, and for the node owners, since they will receive the payments anyway.

This scenario is indeed possible, but it is assumed that since tax payments are supposed to be made in small increments and automatically, the majority of clients won't be interested in this fraudulent scheme.

13.4 Loss of Wallet

A wallet is just a text file, which can easily be lost by it owner.

This indeed is possible, but is not a risk at all, since all wallets are maintained by the network and anyone can easily pull any wallet back. The only sensitive part is the private key of the wallet. If that file is lost, the wallet can't pay anything anymore. This is the file all wallet owners must keep safe.

14 Conclusion

To be written...

15 Acknowledgements

The first version of this document was created by Yegor Bugayenko in May 2018. Since then there were many other contributors, who helped to make the text better, corrected mistakes, suggested and made improvements. This is a non-complete list of the most active participants of this collaborative

process (in alphabetic order of their first names): Andrey Valyaev, Michael Silver, Soh Yuan Chin.

References

Andreessen, Marc (2014). "Why Bitcoin Matters." *New York Times* 21.

Back, Adam (1997). *Hashcash*.

Bohr, Jeremiah et al. (2014). "Who uses bitcoin? an exploration of the bitcoin community." *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*. IEEE, pp. 94–101.

Boyen, Xavier et al. (2016). *Blockchain-free Cryptocurrencies: A Framework for Truly Decentralised Fast Transactions*. Tech. rep. Cryptology ePrint Archive, Report 2016/871.

Buterin, Vitalik (2013). "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." URL: https://github.com/ethereum/wiki/wiki/White-Paper.

Cheah, Eng-Tuck et al. (2015). "Speculative bubbles in Bitcoin markets? An empirical investigation into the fundamental value of Bitcoin." *Economics Letters* 130, pp. 32–36.

Everaere, Patricia et al. (2010). "Double Spending Protection for E-cash Based on Risk Management." *International Conference on Information Security*. Springer, pp. 394–408.

Fekkes, Lotte et al. (2018). "Comparing Bitcoin and Ethereum."

Foroglou, George et al. (2015). "Further applications of the blockchain." *12th Student Conference on Managerial Science and Technology*.

Hayes, Adam S (2017). "Cryptocurrency value formation: An empirical study leading to a cost of production model for valuing bitcoin." *Telematics and Informatics* 34.7, pp. 1308–1321.

Iwamura, Mitsuru et al. (2014). "Can we stabilize the price of a Cryptocurrency?: Understanding the design of Bitcoin and its potential to compete with Central Bank money."

Karame, Ghassan O. et al. (2012). "Double-spending Fast Payments in Bitcoin." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, pp. 906–917.

Kaskaloglu, Kerem (2014). "Near Zero Bitcoin Transaction Fees Cannot Last Forever." *The International Conference on Digital Security and Forensics*. The Society of Digital Information and Wireless Communication, pp. 91–99.

Kiayias, Aggelos et al. (2015). "Speed-Security Tradeoffs in Blockchain Protocols." *IACR Cryptology ePrint Archive* 2015, p. 1019.

Möser, Malte et al. (2015). "Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees." *International Conference on Financial Cryptography and Data Security*. Springer, pp. 19–33.

Nakamoto, Satoshi (2008). "Bitcoin: A Peer-to-peer Electronic Cash System."

Pilkington, Marc (2016). "Blockchain Technology: Principles and Applications." *Research Handbook on Digital Transformations*, p. 225.

Popov, Serguei (2017). "The Tangle." URL: https://iotatoken.com/IOTA_Whitepaper.pdf.

Van Alstyne, Marshall (2014). "Why Bitcoin Has Value." *Communications of the ACM* 57.5, pp. 30–32.