Blockchain engineering

Digital Euro Team III

March 2022

1 Introduction

The recent developments in the world showed that digital cash is not only a theoretical idea but a practical need. To roll out such a dramatic change in a society consisting of millions of people, such as the EU, it needs to clearly demonstrate its advantages over the current technology. Consequently, the presence of a severe downside could seriously block the potential adoption. Hence, our team tries to tackle one of the most serious problems with digital cash – the double-spending problem. First, some of the current methods to prevent and detect it are discussed. Then the requirements and the assumptions about our implementation are introduced. Following that, the actual protocol is described in more detail. In the end, a short discussion about the potential consequences of such a system is presented.

2 Related Research

2.1 Prevention

2.1.1 Online

There are different types of proposals to deal with the prevention of the double-spending problem in the online setting. The first one is to use blockchain such as Bitcoin [9]. The easiest way to be sure that the other party will not double-spend in Bitcoin is to wait long enough after the receiving transaction has been put in the long-term chain. The biggest downside of such a strategy, which makes it unusable for e-cash, is the waiting time – most of the use cases of e-cash, such as grocery shopping or paying a bill in a restaurant, require nearly instant confirmation. Thus, zero-confirmation transactions have been a popular research topic. One of the first attempts to deal with those were the so-called green addresses, but those have not succeeded. However, parts of the idea about trust could be reused by us. Other methods include broadcasting the transaction to a set of neighbors or a random set of nodes [10]. Another family of solutions to the double-spending problem was to use some form of centralization. For instance, that's what was used by David Chaum in his e-cash scheme in order to deal with double-spending.

2.1.2 Offline

The main problem of the schemes previously presented is the online restriction itself. The assumption that a user will always be online is not realistic, considering that a disaster is possible. Chaum realized that an entirely online solution to the problem will not actually solve it completely, thus, his crypto-cash also included a detection mechanism. He used cryptography to enable the “victims” of double-spending to reveal the identity of the double-spender with high probability. Such a scheme was adopted by many researchers in the field [7] [3] [5]. The problem with all of these is that they rely on the legal system and banks to deal with the fraud, which is a constraint our team wants to avoid.

A different approach to tackle the problem was the use of secure tamper-proof hardware. The device was in the form of a smart wallet or card, which kept track of the balance of the user and updated it if the user received/spent money. The method was adopted by many companies, some of the big ones being MasterCard and Visa. Unfortunately, the technology never succeeded. Interestingly, the main reason for failure was not the actual security of the devices. Similar to physical cash, when you lose the hardware or it gets damaged, the holder loses his money. In addition, you can trade only with people or institutions who use and accept that particular technology.

2.2 Detection

Detection of double spending can be done in two particular stages of the transaction: before the transaction is performed, known as proactive detection, and after the transaction has happened, which we know as retroactive detection.
Proactive detection happens before the transaction is registered, during the chain verification phase. In order to perform this, we need some previously collected of the malicious node and its transaction history. This either happens by having a previous transaction with the malicious node or, more likely, through information dissemination through the network.
Retroactive detection happens after the transaction. Via dissemination, we obtain new information regarding sequence numbers, which shows us a double spending event has occurred.

2.2.1 Online

In a centralized way, for example with Bitcoin, we have lower speed than is usually required in smaller transactions. The usual way of dealing with these fast transactions is accepting the risk of a payment not being approved, which will be the problem of the receiver. It is mitigated by waiting till the payment has been propagated through the network. The receiver monitors a random sample of nodes and waits to see if its payment is occurring in the network, detecting if the payment has actually been accepted. If we take a large enough sample to monitor, we can prevent the majority of double spending. [6] [1]

An idea to stop double spending by malicious nodes in a pair-based ledger is to anonymize requests to view a ledger. This way, the malicious node cannot handcraft a faulty chain, since it does not know what the node knows it is trying to cheat on. The intermediary anonymizer nodes would shield the communication between the two nodes. It still poses the risk that the anonymizer node itself is malicious. In order to guarantee fairness, the systeem would require on anonymizer nodes to be audited. [8]

2.2.2 Offline

In general, there seems to be little research done on double spending detection in an offline environment. One of the systems proposed is an off-line karma system, where tokens need to be reminted after a certain time. [4]

3 Requirements

In order to bound our solution space, we need to identify some constraints. Aside from working with TrustChain, we have identified some other constraints that influence our design space. A list can be found below:

- Fully offline capable
- Completely distributed
- Permissionless
- Pseudo-anonymous
- Independent of other authorities - legal, bank

Given the research in the previous section and the constraints, we have decided to stick to double-spending detection, as it is most feasible in an offline environment. Double-spending prevention is generally more tricky even in an online requirement, as it also requires some waiting for dissemination of the transaction through the network. Only then can the purchased goods or services be delivered.

In short, we want to be able to share trustworthiness across the network, allowing us to check if our peers are known for previous good behaviour, in essence known not to double-spend. If we see others validate them upon the exchange of trust scores, the user can be indicate of their reliance within the network. This in turn allows the user to make a more informed decision regarding sending or receiving money from this peer.

To summarize the must-haves:

- Allow storage of key-value pairs with public keys of peers
- Allow users to share their knowledge of others during interactions
- Communicate these scores to the user when interacting with a score with a wallet

- Create some form of computation towards a trust score

And the wont-haves:

- Definitive formula on how to calculate trust based on the transaction graph and previously known attacks
- Blocking of users from the network

4 Protocol

4.1 Implementation details

Our initial basic setup allows us to create signed key-pairs in a JSON file. This consist of the public keys of wallets, together with a score between 0 and 100. Here, 100 symbolizes maximum available trust. Upon creation, records are initialized on the JSON file.

Upon trying to complete a transaction, we go through our collected records and see if we can find a matching public key of the peer we are interacting with. If we cannot find this, the user is informed that we currently have no information on this user. Alternatively, if we do find a key-pair, we read the trust score from the database. We define some thresholds, as can be seen in Figure 1, where we give a color indication based on the value associated with the key. How this would look to the end user is shown in Figure 2.

After interacting with another user, the receiver gets a list of the last 50 users the sender has interacted with. If there are familiar users within this list, we can update their trust scores. For now, this is simply done by incrementing their trust score by one percentage point. If we encounter a new public key within the sent list, we add this person into our database. Over time, this will allow us to see nodes within the network that thoroughly interact with other users, which we intuitively trust more than users with limited interaction.

5 Discussion

Although the systems sets out what was intended by us from a technical viewpoint, it still suffers from some difficult problems. Further research will have to answer these problems. It does however show technological capability to propagate trust through the system.

5.1 Reputation scores

One issue is that the trust score could lead to a sort of social credit system, similar to Chinese implementations. These could infer social implications, as users might get excluded due to their low trust scores. These social outcasts could theoretically only be accepted by each other, leading to double-spending

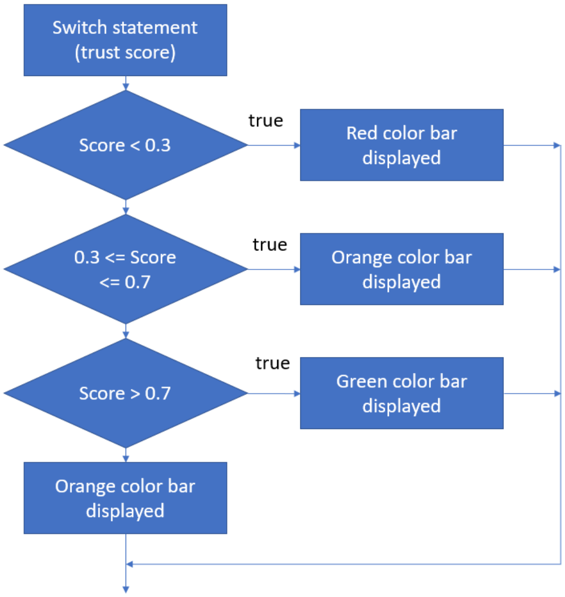
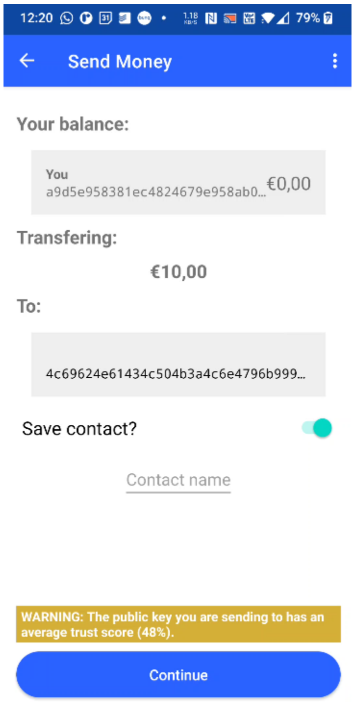

Switch statement (trust score)
Score < 0.3
true
Red color bar displayed
0.3 <= Score <= 0.7
true
Orange color bar displayed
Score > 0.7
true
Green color bar displayed
Orange color bar displayed

Figure 1: Flow diagram for displaying information regarding trust scores

regions within the network. This is not desired, as they could be excluded from a majority of their desired transactions. Similar consequences can be observed within credit scores in America. Not meeting a certain threshold can exclude you from lending or general business transactions.

5.2 Security concerns

Other issues are more closely related to the security of these trust scores. A high trust score could become a desirable trait, since more people would trust the user to do business with them. Once this happens, people will try to game the system. One possible adversary strategy would be to boost you own scores by repeatedly transfer between two wallets (or a similar cycle within a graph), resulting in a high trust score. Similar approaches can be seen in other fields, such as search algorithms [2]. For now, the algorithms creating such scores are generally hidden and in constant change. This security by obscurity approach could be also applied to the trust score algorithm. By not using a linear function, but for example machine learning, we could obfuscate some of the factors that the algorithm considers predictive for double-spending. If these factors remain unknown, bad actors have more troubles understanding and manipulating the system versus a publicly known algorithm. A major downside of security by obscurity is that once the method of calculating trust scores is known, it can be easily taken advantage of to boost your own score.



Your balance:

You
a9d5e958381ec4824679e958ab0... €0,00

Transfering:

€10,00

To:

4c69624e61434c504b3a4c6e4796b999...

Save contact?

Contact name

WARNING: The public key you are sending to has an average trust score (48%).

Figure 2: Screen capture of a transaction showing a trust score

References

[1] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. Have a snack, pay with bitcoins. In *IEEE P2P 2013 Proceedings*, pages 1–5, 2013.

[2] S. Bradshaw. Disinformation optimised: gaming search engine algorithms to amplify junk news. 2019.

[3] Chun-I Fan, Vincent Shi-Ming Huang, and Yao-Chun Yu. User efficient recoverable off-line e-cash scheme with fast anonymity revoking. *Mathematical and Computer Modelling*, 58(1):227–237, 2013. Financial IT & Security and 2010 International Symposium on Computational Electronics.

[4] Flavio D Garcia and Jaap-Henk Hoepman. Off-line karma: A decentralized currency for peer-to-peer and grid applications. In *International Conference on Applied Cryptography and Network Security*, pages 364–377. Springer, 2005.

[5] Xu Danhui Kang, Baoyuan. Secure electronic cash scheme with anonymity revocation. *Mobile Information Systems*, 04 2016.

[6] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. Cryptology ePrint Archive, Report 2012/248, 2012. https://ia.cr/2012/248.

[7] Rabin T. Krawczyk, H. Chameleon signatures. *Symposium on Network and Distributed Systems Security, NDSS'00, 2000, pp. 143–154*, 2000.

[8] Umeer Mohammad. Enabling double-spending detection in a pair-based ledger. 2019.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[10] Cristina Pérez-Solà, Sergi Delgado-Segura, Guillermo Navarro-Arribas, and Jordi Herrera-Joancomartí. Double-spending prevention for bitcoin zero-confirmation transactions. *Int. J. Inf. Secur.*, 18(4):451–463, aug 2019.