

DELFT UNIVERSITY OF TECHNOLOGY

THESIS PROJECT

Thesis Motivation Document

by Umeer Mohammad

1 Motivation

The TrustChain is a protocol for a share interconnected data structure with enormous potential, it is designed to record transactions among strangers without central control, and it is able to support high transaction volumes. This system is highly scalable while maintaining stable performance independent of the numbers of active users, which is a hard to achieve in blockchain structure where there is a unique shared ledger [1].

This is enabled by a network composed of individual semi-independent ledger (every node has its own) and rules that define how these are constructed.

A transaction between two parties creates a new element in both the participant's ledger and this new element contains some pieces of information like the transaction itself, details of the previous element in the ledger and signatures of both the parties.

A double spending attack allows the attacker to utilize resources already consumed before, allowing him to fraud two or more node with the same resource set.

Let's consider a real-world example to better understand the problem.

You decide to go to a restaurant to order a pizza worth 10 euro, you pay in cash and the note goes into the shop's cash desk. Now by no means you can spend that note somewhere else that day. This constriction is easy to understand in the real world, but in TrustChain things are a bit different, here it is possible to double spend your 10 euro somewhere else multiple times before being caught by a cashier, and this might take quite some time.

This is possible because the defence system against these attacks is effective but has a weak point. Soon or later an adversarial node will enter in contact with a node that has knowledge of his previous transactions, in such a situation attempting a double spending attack will result in failure. In a low-entropy transactions system, behaviour such as this can take a long time before being detected and this can result in chains correction increasingly complicated.

Like in any good idea, there are weak points that need to be addressed and re-enforced in order for it to become a commercially approved technology. With my thesis project, I want to approach one of these weaknesses with the goal of finding a better solution for it.

2 Solution

The concept behind the solution relies on being able to ask and check a node chain before and after transaction in anonymity, this will limit his possibility to double spend.

Having the identity of the requester node hidden during the verification phase is the key, because an adversarial node will not be able to build and send back a custom chain¹ that can convince the receiver, or at least not with a sustainable probability of success.

The solution is composed of three components:

- Component A:

An auditing system using anonymizer nodes. The function of this node is to be the relay of messages for nodes that want to stay anonymous to the recipient, in such a way that the recipient will not get to know about the original identity of the sender, and so will not be able to send back node customized replies. This component plays a key role in the two remaining components of the solution.

- Component B:

A node X after a transaction with Y, using multiple² anonymizer node has to periodically check the Y chain and looks for incongruence. This operation is repeated until the transaction is old enough³. The scope of this operation is to catch nodes that try to hide or modify blocks of their chain.

If a node finds an incongruency, it is rewarded but if the node refuses to do the proper checking and an incongruency is detected later on, the node is penalized and considered part of the illegal activity. However if the node Y goes offline, it became impossible to performing these checking, this problem is partially addressed by the next step.

- Component C:

Every time a node performs the checking described in Component B, it needs to keep a log of the node checked, along with a time stamp. By mean of this, all the nodes that have done a transaction with Y will have a copy of his chain which can be used in two scenarios.

1) A new node right before starting a transaction with Y can ask for his chain and compare it with chains collected by other nodes is his transaction history. This can guarantee that Y is not able to modify his chain just right before he is about to do a transaction (even if we use anonymizer node Y can guess the origin of the request base on time of the request).

2) If Y stay offline for a long period, other nodes are still able (to some extent) to perform Component B using the logs.⁴

The evaluation of the solution will happen in three phases, initially, a good representative model of TrustChain will be built on OMNeT++ and it will be tuned up until the simulation and the real implementation behave in the same way. So design, evaluation and development of attack vectors will be done and implemented in the simulator, from where data will be collected with multiple simulations (with different parameters of network/attacker) in order to be used as reference base point.

Finally, the solution set above will be implemented on the simulator and tested intensively, the collected data will be compared to the reference one in order to study the improvement results. This last phase will be iterated with many solution's parameter's settings until an optimal solution is found.

¹The adversarial node can have two or multiple node with all the transaction that can be switched depending on the requester identity

²Using multiple anonymizer node allows to do compare the results of the auditing so that it is possible to be resilient to adversarial anonymizer (if >50% of the node used are honest)

³This value should be large enough to guarantee security and practicality at the same time

⁴This specific case might need a better solution like the use of alias/custodian

3 Time Plan

Task/Goals	Deadline
Background reading	1 January
First meeting	11 January
Set-up workspace	18 January
First draft of problem&solution doc	25 January
Getting familiar with OverSim	25 January
Speak with TrustChain developers	February
Finalize solution on paper	March
Developing prototype on simulator	April
Implementation of the solution on simulation	July
Data collection and conclusion	7 August
Study on paper optimization	9 August
Complete thesis introduction	16 August
Implementation of the optimized solution	23 August
Run simulations and data collection	30 August
Completed fist draft of thesis	20 September
Starting creation of presentation	27 September
Finalization of thesis&presentation	4 Oct
Test presentation with colleagues	4 Oct
Thesis defence	18 Oct

References

- [1] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, thirdquarter 2016.