

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 22, 2018

J. Pouwelse, Ed.
Delft University of Technology
January 18, 2018

TrustChain protocol
draft-pouwelse-trustchain-00

Abstract

TrustChain is a protocol for a networked datastructure, designed to act as a trust ledger. This protocol acts as a decentral alternative to platforms like eBay, Airbnb, and Uber. It is specifically designed to record transactions among strangers without central control, support high transaction volumes, be application neutral, and avoid vendor lock-in. The protocol defines recording transactions in an ordered list using an append-only datastructure, a new communication overlay, and a horizontally scalable consensus protocol based on checkpoint consensus, called CHECO. Trustchain has resistance to traditional blockchain attacks, such as the 51 percent majority attack. This is achieved by using a graph-based append-only datastructure combined with a personal blockchain for each participant with their own genesis block.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 22, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Purpose	4
1.2.	Requirements Language	4
1.3.	Terminology	4
2.	Trustchain Stack: Engineering trust	5
3.	Trustchain Fabric: internal data structure	7
3.1.	Architecture	7
3.2.	TxBLOCK specification	9
3.3.	Asynchronicity	10
3.4.	CHECO: Consensus protocol and block format	11
4.	IPv8: Overlay for identity, discovery and trust	11
4.1.	Identity establishment and discovery	12
4.2.	Attestations and trust	12
4.3.	Peer-to-peer cryptographically signed messaging	13
4.4.	NAT traversal	13
5.	Attack resistances	14
5.1.	Sybil attacks	14
5.2.	Double spending attack	15
5.3.	Replay attack	15
5.4.	Whitewashing attack	15
5.5.	Spam attack	15
5.6.	DDoS	16
6.	Acknowledgements	16
7.	IANA Considerations	16
8.	Security Considerations	16
9.	References	17
	Author's Address	20

[1.](#) Introduction

For the past 10 years various distributed ledgers have been deployed and used. This protocol aims to establish some form of trust using software.

Creating trust between strangers is at the core of numerous successful Internet companies. Starting 22 years ago, Craigslist

offered an unmoderated mailing list of advertisements and gossip on which buyer and seller could be trusted. eBay formalised this in 1997 and introduced a star-based rating system that enables traders to build a trustworthy profile [[resnick2002trust](#)]. The e-commerce platform was launched at a time when people were still hesitant to use their credit card on a technology called The Internet. Nowadays, people let strangers sleep in their houses using Airbnb (since 2008). We trust Uber (since 2009) with our physical security and get into cars late at night with a driver that has not undergone a criminal background check or given a government license.

The Trustchain protocol aims to create a generic approach and continue this evolution of building trust. Compared to successful central platforms we propose a distributed open underlying infrastructure, based on blockchain inspired technology. Bitcoin created money without the need for banks [[nakamoto2008bitcoin](#)]. In the past, people were required to trust a central bank and a host of other intermediaries when making payments [[kokkola2011payment](#)]. The fundamental technology of Bitcoin, blockchain, radically reduced the need to trust financial middlemen. It bootstrapped an economy where no one can be stopped from spending their money. Despite widespread speculation and ecosystems being worth billions, blockchain in general suffers from scalability issues due to inefficient mechanisms for fraud prevention. Bitcoin is theoretically limited to seven transactions per second and Ethereum has a throughput of around 20 transactions per second [[vukolic2015quest](#)]. Despite various scalability efforts like proof-of-stake and sharding, broader adoption of blockchain stays out. Mt. Gox was at one point the largest Bitcoin exchange worldwide. While a majority of Internet users trust the company behind popular platforms, the events involving Mt. Gox highlighted how digital trust can be established and compromised [[mcmillan2014inside](#)]. In 2014, hackers stole Bitcoin, worth around \$460 million at that time. This event, together with major data breaches in 2017 at high-profile companies like Uber and Equifax, exposed the weakness of centralized architectures [[apostle2017uber](#)]. These events motivated this proposed protocol around decentralised infrastructures, not owned or operated by a single authority. The generic problem of building trust between strangers resides on the edge of technology, sociology and behavioural science [[yan2008trust](#)]. The question whether someone can be trusted, depends on properties like personality, level of authority, culture and past behaviour. In this protocol, we address the trust problem from a technological perspective, using tamper-proof interactions on a scalable blockchain. This structure is built to help ease the detection of fraudulent behaviour and misrepresentation. Trust calculations are out of scope of this work, we provide the enabling mechanisms.

1.1. Purpose

This draft describes the TrustChain architecture, protocols and the used technologies, designed to model application neutral trust between interacting parties. TrustChain relies on a new communication layer on top of existing communication networks which is designed with carrier-grade NAT infrastructure in mind, as well as the network protocol based on the CrawlRequest and TxBlock message types. A consensus protocol called CHECO (Cong et al, 2017 [[cong2017blockchain](#)]) is incorporated into TrustChain, which will be discussed but not elaborated in this draft. It is based on the blockchain paradigm where the complete network represents a ledger where agents' transactions infer an amount of trust between the involved parties, as is described by pouwelse, 2017 [[pouwelse2017trustlaws](#)].

As protocols have slowly been moving towards the business layer in the past decade, TrustChain is implemented on top of a networking overlay and as such is network agnostic. Other examples of moving networking to the business layer are: R3 Corda (Brown, 2017 [[brown2017introducing](#)]) and IOTA (Atzori, 2016 [[atzori2016blockchain](#)]). The overlay, audaciously called IPv8, provides encrypted communication between public keys. This overlay has integrated NAT puncturing to support, for instance, Android-to-Android overlay communication, does not require any central server, lacks central authorities, and can run directly on top of UDP, TCP, or other protocols. As such, IPv8 provides a set of communication methods and messages that provide the required functionality to let TrustChain function properly on both PC networks or smartphone-only networks.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

1.3. Terminology

Identity

The actual user representation, but can not be directly used, since all information is identifier based.

Identifier

A reference that is owned by a given Identity, referring to this Identity. Any Identity can have multiple Identifiers, whilst staying anonymous.

Agent

A node in the TrustChain network representing an identifier for a given identity.

Message

The basic unit of TrustChain communication. A message will have different representations on the wire depending on the transport protocol used. Messages are typically multiplexed into a datagram for transmission.

Datagram

A sequence of messages that is offered as a unit to the underlying transport protocol (UDP, etc.).

Transaction

An interaction between two agents containing information on both parties and what has been transacted. This is application agnostic, meaning that any given application can infer what type of information it needs based on a collection of transaction.

Signature

A cryptographic function that used the private key to create a representing string that can be verified by any other party using the signer's public key.

Hash

The result of applying a cryptographic hash function, more specifically SHA-256 [[FIPS180-4](#)], to a piece of data.

2. Trustchain Stack: Engineering trust

Our principle mechanism to establish some form of trust is: if everyone keeps their secret keys secure, then no signed transaction can be spoofed on the overlay by any significant likelihood. We refer to the Trustchain protocol when discussing the mechanism to record interactions which are cryptographically signed by multiple parties. We explicitly do not support transactions signed by only a single party, which is the foundation of Bitcoin. Our foundation is a multi-signature agreement, without mono-signature support. The Trustchain stack refers to the full system which also includes the upper application layer, the network overlay and self-sovereign identity layer. Together they form a complete solution stack.

The concept of a Self-Sovereign Identity (SSI) (Abraham, 2017 [[abrahamself](#)]) means that agents have full control over their identity data, and provide it to those who need to validate it, without relying on a central repository of identity data of any kind. A large part of any SSI based system is rooted in the problem of

proofs and attestations: proof can be anything, such as a secret message that is re-encrypted with a shared secret. But the most challenging part is working with attestations, an attestation is when a third party validates that according to their records, the claims are true, a more transitive property of trust: C attests to B that A is who it claims to be, Azouvi et al, 2017 [[azouvi2017secure](#)]. As such, an attestation from the right authority could be more trustworthy than a proof, which might have been forged. However, attestations can be a burden on the agent as the information can be sensitive, the information needs to be maintained so that only specific agents can access it. Our stack does not constrain the choice of SSI system, but our implementation is focused on the Boneh-Franklin [[boneh2004secure](#)] 2-DNF scheme ("Evaluating 2-DNF Formulas on Ciphertexts").

Based upon the assumption that these identities are persistent and secure, the new architecture (or Fabric) is designed to use Peer-to-Peer communication to increase the transaction throughput. This communication is based on the new networking overlay: IPv8, which handles peer discovery, making connections with them across NAT boxes and peer-to-peer cryptographically signed messaging. As IPv8 is transport and application agnostic, it can run over any transport protocol: it does not depend on IP and may run on top of NDN, XIA, and other new Internet architectures.

To ensure that the blockchain always is in a valid state, a new horizontal scaling consensus protocol is proposed: CHECO. CHECO is specifically designed to counter the vulnerabilities that a distributed, permissionless, multi-chain architecture will have to cope with (although this also creates innate vulnerabilities to other kinds of attacks). By creating an indication of the state of validity for each agent, the responsibility of verification lies with the agent itself, a malicious agent in an invalid state can easily be detected, and should be avoided for interactions.

We do not constrain or limit the applications utilising this blockchain, each transaction block can contain information of arbitrary content and length. Currently there is a basic decentralised market implemented. More specialised market have also be implemented and emulated such as a open Taxi service market and a mortgage investment market. Multiple applications can be used at the same time, next to the decentralised market, for instance: it could be used for byte accounting in a ad-hoc Manet [[jethanandani2017accounting](#)].

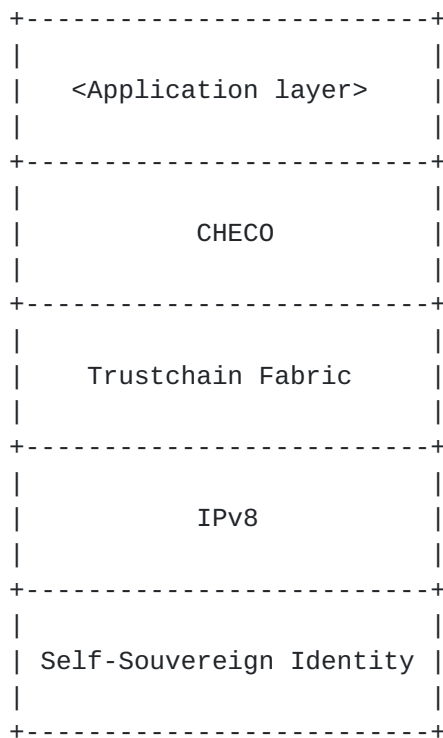


Figure 1

3. Trustchain Fabric: internal data structure

Trustchain is designed to be a non-blocking format for agents that supports simultaneous interactions with other agents. Non-blocking is a requirement rooted in the immutability of the chain and the strict ordering of the blocks. To support this, the blocks are designed to be dependent on signing by all participating agents, and will be called TxBlocks henceforth, as is described in this section, along with the macro data structure in which these blocks are used.

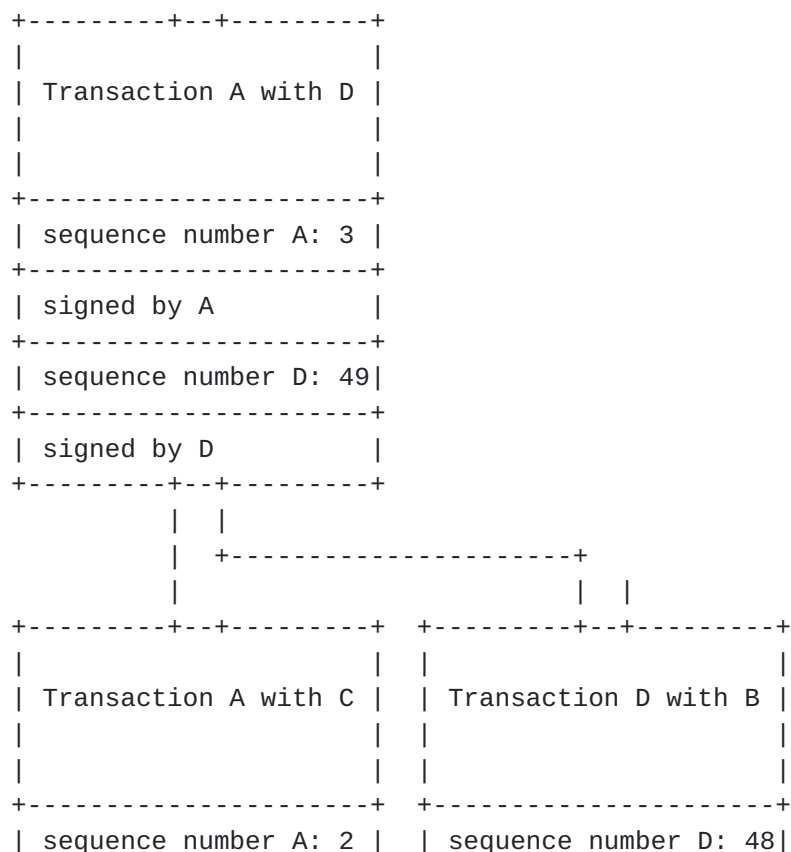
3.1. Architecture

In contrast to traditional blockchains, in Trustchain every agent in the network has its own genesis node, in essence creating a personal blockchain for each agent. Each interaction creates a new transaction block which is based on the last block of the two (or more) concerned parties. This does not only influence the block-creation speed, but also the amount of effort needed to verify a chain. Along with some other security properties, this is one of the implicit capabilities of this protocol.

By removing the proof-of-work mechanism needed for classic blockchain implementations, Trustchain yields inherent horizontal scalability.

However the cost of scalability is that each application requires a mechanism to guard against transaction spam and abuse. Trustchain is based on the assumption that both parties agree on the transaction before signing it, making tampering inherently easy to detect. One of the aspects that supports this is the fact that TrustChain is organised as a set of temporally ordered, intertwining chains, which form a Directed Acyclic Graph (DAG). This is called a "bottom-up consensus model", giving the participants the responsibility to verify the correctness of the transaction in stead of a central (sometimes randomly chosen) elected leadership.

Trustchain depends on signatures from all participants in a transaction, creating a n-to-n node. This system is extendible, as mentioned before, by extending the transaction description to provide specific properties. Each transaction is stored in a block, in agreement-block format, each block has parts that are signed and submitted by all participating parties, where the initial request is called the block-proposal and a completely signed and validated block is called a agreement-block (where pair indicates the cooperation, not the limitation to two participating parties). These are signed and sequenced so that each sequence number is unique in its accessory chain, as can be seen below in the general structure diagram.



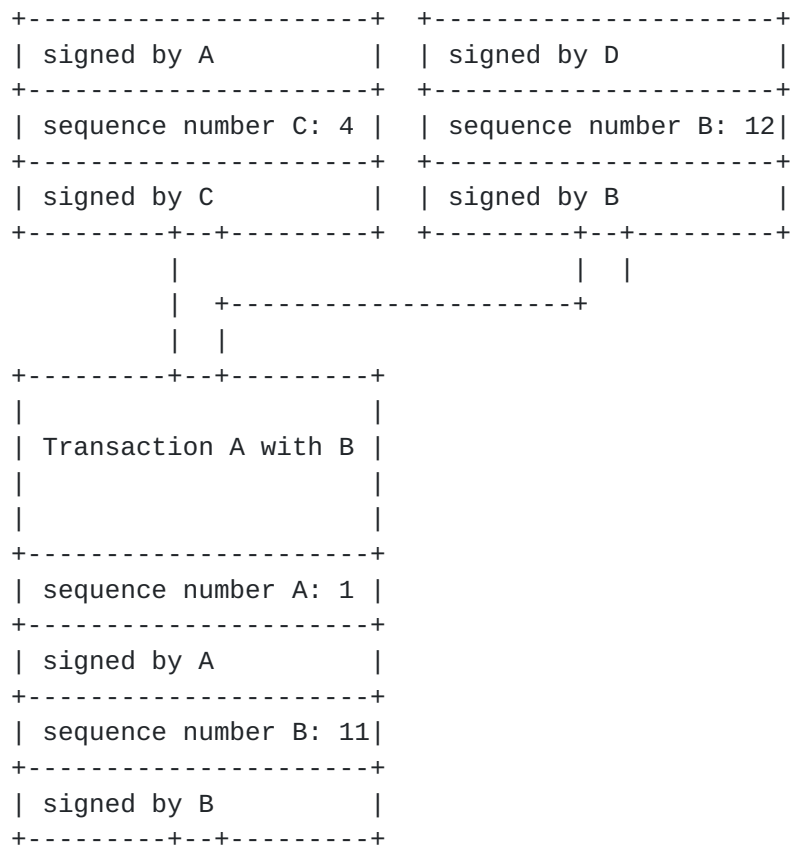


Figure 2

3.2. TxBlock specification

Using the proposal-block and agreement-block formats means signing the blocks on the current views of the respective parties: the requester and the responder(s). Each party signs and fills the block with the information that it has at that specific point in time. The requester fills the structure with his own previous hash and his own part of the transaction data, signs it and sends it to the responder(s), which in turn construct the other sections of the block, if it agrees with the content before sending it back. This nullifies any ordering and asynchronicity issues, since the requester constructs the block with the information that he has, and keeps it in memory while it waits on the responder to send the finished block back.

Number	Description	Type	Size (bytes)
1	Requester public key	Char array	74
2	Requester sequence number	Unsigned int	4
3	Responder public key	Char array	74
4	Responder sequence number	Unsigned int	4
5	Requester previous hash	Char array	32
6	Signature	Char array	64
7	Transaction block size (n)	Unsigned int	4
8	Transaction block	Char array	n
	Total:		256 + n

Table 1: TxBlock fields description

3.3. Asynchronicity

Because there is the need to communicate between the requester and responder(s), there will be a delay which may be significant. To have a high level of asynchronicity and enable multiple peers interacting simultaneously, extending the chain should be able while waiting for a response. In order to do this, the block refers to the previous block using the hash of the requesters part, since this is the only stable reference at that point. The other hash reference (the "previous hash responder") can then be either the "hash requester" or "hash responder" part of the head-block of the responder chain. Which one is used depends on whether the responder was the requester or responder in its previous interaction. This mechanic is also used for the "previous hash requester" field, but this reference is known when the block is created. In effect, this results to theoretically unlimited horizontal scalability: the more actors are active on the chain, the more throughput can be achieved. Though this is in fact limited by the memory speed, or database slowdown when the chain grows.

One of the drawbacks of this mechanic is when the responders does not sign and respond, whether because it will/can not, there will be an orphan block. While this is not a vulnerability on itself, it might be the starting point of a certain type of attack (the other "normal" types of attacks used for blockchains can be mitigated, at least to a certain level, as is described in resistances ([Section 5](#))). The adversary might let someone initiate a transaction an block creation, after which it will create an orphan. Doing this multiple times in a short time span will force the requester to use a considerable amount of processing power and memory, all the while injecting orphan block into his chain. As mentioned before, this is not a vulnerability in itself, but might be a launchpad for a more elaborate attack. Though

one coping method could be to split larger, more vulnerable transactions up into multiple smaller transactions. This way the consequences stay the same for the malicious actor, but the losses are smaller.

3.4. CHECO: Consensus protocol and block format

The final part of the Trustchain Fabric is CHECO, a horizontally scaling consensus protocol specifically designed for multi-chain implementation and completely application agnostic. CHECO is based on three separate protocols: A consensus protocol, a transaction protocol and a validation protocol, as well as an extension of the architecture by introducing a new type of block next to the TxBlock: CpBlock. Every round a set of so-called facilitators is selected at random, which collect the CP and TX blocks, to feed to the validation protocol, after which the results are broadcast before a new round starts.

CHECO is designed to create an internal state ledger for each chain, without having to rely on separate methods or instances. This is achieved by introducing a new block: the Checkpoint block (CpBlock), which contains a hash pointer to the previous block, along with a hash of the consensus result, round and sequence number and, lastly, a signature. The consensus result is defined as a tuple containing the validity states of the blocks agreed on by the facilitators of that round and the round number. If your chain is deemed valid, a CpBlock is injected, thus validating the state of the chain. While the content of the injected CpBlock differs from the TxBlock, these blocks do not interfere with the transaction protocol, since they fit in the architecture without modifying it.

Instead of requiring a proof-of-work (as seen in more conventional blockchain implementations), CHECO is round based, creating a consensus state every so often, creating a fully asynchronous and horizontally scaling protocol. These facilitators are chosen randomly each round, and will collect the CP blocks from all other nodes since their last CpBlock. Validation is done using Asynchronous Common Subset (ACS) algorithm based on HoneyBadgerBFT (Miller et al, 2016 [[miller2016honey](#)]), a byzantine consensus algorithm.

4. IPv8: Overlay for identity, discovery and trust

To enable this new platform to function properly, a new method to find, connect to and manage agents was needed. Additionally, new models for identity verification, network discovery and inter-peer trust were required to enable these agent methods. It is a network stack, a set of protocols and models, that separates concerns and

enables applications (such as Trustchain) to use the needed methods and protocols, without giving up interoperability and upgradeability. On top of this interface lies the actual TrustChain overlay, which uses the components and protocols of the IPv8 interface to create the specific functionalities needed for TrustChain to function.

IPv8 is built keeping the Unix Philosophy of creating small components that are easy to understand and test. This is mainly why the complete networking system used in Tribler was re-written as a generic networking interface, enabling modifications and additions without losing any functionality. Although agents might use different protocols based on their capabilities, IPv8 is a basic layer over a multitude of networking components and subsystems, creating a means to communicate with other agents regardless of networking capabilities. An open source reference implementation based on Python is available on Github, called py-ipv8. Interoperable open implementations in Java and Kotlin are still only partially functional.

4.1. Identity establishment and discovery

Identities are created, attested and distributed over the Trustchain using IPv8 as the communication interface. These are all public and self sovereign, leading to distributed when an agent creates a new identity, and are organised using a Public Key Infrastructure (PKI). This distribution is done to agents who have (or might have) interest in having the identity of this agent, since they are in the same cluster or due to other factors. Spreading information this way is called Gossiping, since agents learn of new identities, and spreading them among directly connected agents like a gossip would spread.

Discovering identities is done based on Distributed Hash Tables (DHT), somewhat similar to the Domain Name System (DNS) currently used for the web, using Random-Walk and Live-Edge-Walk: discovery protocols for DHTs, respectively based on making random DHT queries in order to learn about a large number of identities quickly and making pseudo-randomised queries about the agents with the highest trust scores in the network. Using the random walks causes the DHTs to converge much faster, whilst having a small load at the very beginning, but by traversing the live edges, TrustChain is more spam and Sybil attack resistant.

4.2. Attestations and trust

The primary problem with identities and proofs is falsification, and need to be verified to prevent this. However, even proofs can be forged, leading to the problem of needing trust in a proving party, which can not be solved by having a more centralised proving party.

This is where attestations are used, witness reports that the identity in question is actually valid, and requires some level of trust between the agents. Attestations are a method to enable agents to validate interactive zero-knowledge proofs (ZKPs) within a network of agents, a so-called web-of-trust [[azouvi2017secure](#)], but the transitive property of trust is used to prevent the need for every agent to verify an identity itself as is noted in [Section 2](#).

In this case, the probabilistic homomorphic asymmetrical encryption scheme of Boneh-Franklin [[boneh2004secure](#)] is used to validate these proofs, meaning a form of randomness is used in the encryption and computations on the ciphertext result in a valid plaintext. Using Boneh-Franklin leads to these ZKPs to be hardened against chosen plaintext attacks (an attacker can encrypt the suspected plaintext to see if the ciphertexts match) with the probabilistic aspect. These attestations are tied to metadata, which can be verified separately and the related identities are stored internally in a database, and these metadata attributes are gossiped around the network using Trustchain.

[4.3.](#) Peer-to-peer cryptographically signed messaging

Most messages that are sent between peers are encrypted, the keys are retrieved from either the PKI, and verified based on the key attestations, or from the internal database after verification already has been done. Encryption is done using the ECDSA encryption scheme [[johnson2001elliptic](#)], which is an Elliptic Curve based cryptography system. This enables IPv8 to send messages effectively to a multitude of agents when (at the least) their identities are known, preferably also attested for. Having such a low bar to encrypt the messages discourages the use of unencrypted communications channels, making interactions secure almost as early as the handshake, as the first message sent to any agent can be already encrypted with its respective key (which is retrieved from the database, other agents or the initial bootstrapping list).

[4.4.](#) NAT traversal

Network Address Translation (NAT) is omnipresent in the modern Internet, mostly due to networks being separated and the limited amount of global IP addresses available. Most consumer devices are behind a number of layers of NAT, but data center nodes can be behind NAT for security or virtualisation reasons. Containerised deployments are making things worse, making every peer based communication scheme must have a way to traverse NATs, otherwise operations will be affected. Even nodes meant to run with real IP addresses must implement NAT traversal techniques, as they may need to establish connections to peers behind NAT. Messages puncturing

based on UDP is key to this overlay. It conducts a random network walk to preserve connectivity under churn. Participants helps each other to puncture NAT infrastructure. Each participant will periodically introduce and connect some of its neighbors. When their random neighbors do not yet know each other, a new participant is discovered. Carefully timed concurrent UDP messages are used to traverse carrier-grade NAT infrastructure. Implementation, deployment and measurements of smartphones users has show that it is possible to build a healthy overlay without servers, even if nearly 100 percent of users are behind a NAT (Android-to-android overlay [[TUDelft2018trustchain](#)]).

5. Attack resistances

For blockchain implementations, attack resistance is an important requirement, especially with horizontal scalability. Therefore it will have to cope with the same difficulties and attacks that other blockchain implementations have to, but due to the novel structure are be countered. With this novel structure, validation, uncertainty, and tamper proofing can be handled in a more intuitive manner while throughput does not need to suffer.

5.1. Sybil attacks

One of the most difficult attacks to repel for a blockchain implementation is the sybil attack, where many agents are injected into the chain (and authenticity cannot easily be verified) to subvert a large portion of the systems voting power or trust (see [section 3.3](#)). Usually peer verification is used to resist this kind of attacks (for instance: proof-of-work), usually resulting in slow systems. But when the influence of the attacker is large enough, even these methods will not be able to stop such an attack.

Trustchain deal with this problem by having an inherently different structure, where each peer has its own origin. On top of that, transaction injection can only be done with two valid signatures, meaning a sybil attacker can only create trust with himself. This results in a network of disconnected agents that have no relation outside of its own cluster, which can easily be identified. Even when the sybils acquire some degree of trust outside of their cluster, using accounting mechanisms the profit from such an attack can only be weakly profitably beneficial with bounded profit (using Netflow, not discussed in this paper) (Otte et al, 2017 [[otte2017trustchain](#)]).

5.2. Double spending attack

Using control over the blockchain to create a fork and creating two different transaction branches is called double spending. This kind of attack can be applied with relative ease to single chain implementations of the blockchain by injecting two conflicting transactions at the same time. Trustchain counters this kind of attack by having the chain verified with each CHECO round, during which the hidden transaction can be easily found. By broadcasting both blocks as a proof-of-fraud the malicious agent will have decreased trust and can be blacklisted or refused service.

5.3. Replay attack

Using the transaction signature of the counterpart, a malicious agent can try to replay a transaction on the blockchain, which results in increased trust or could be used to gain credits. CHECO and the novel structure make it trivial to find the conflicting blocks when verifying the counterparties chain. The two blocks with the same outgoing pointer together make the proof-of-fraud which then can be used to decrease the trust in the malicious party and can be blacklisted or refused service.

5.4. Whitewashing attack

Abusing the permissionless structure of Trustchain to create additional identities at any given point can negate the effect of having trust, so this kind of attack differs from a sybil attack. When an agent suffers from reputation loss, it can simply discard his current identity and take on a new one. Since refusing service to agents with little trust will affect usability and willingness to join the network, an adequate solution can be prioritising strategies. One method for implementing this is described in the paper discussing Netflow (not discussed in this paper) (Otte et al, 2017 [[otte2017trustchain](#)]).

5.5. Spam attack

Since the TxBlocks have a dynamic size, spam (in the sense of useless data in the transaction field) can be used to clog an agent's network or database with excessively large messages, slowing down its operations or bringing them to a complete halt due to memory/network being full. This kind of attack can be coped with by having a throttle per connection to keep some bandwidth available and a limit on the size of the message. If large messages are to be expected, a file based buffer will enable large message transfer without exceeding the memory capacity.

Another type of spam can be identified as a collection of useless messages, clogging the network and database with large amounts of empty messages, which is possible since transaction do not require an agent to pay transaction costs (as BitCoin does). Though this is easily countered by not accepting those messages, leading to the malicious agent having a lot of orphan block.

5.6. DDoS

When massive quantities of useless or empty messages are sent over a network, it might get congested, leading to dropped operations, network congestion and unreachability of agents, or in the most extreme cases even to system failure due to overload. Due to the distributed, peer based communications architecture, this is not feasible without flooding the network of the malicious agent itself, as it has to send messages to each target individually.

6. Acknowledgements

We very much thank the European Union for providing us the required funding for this work. Through EU FP6 and FP7 funding instruments we have been developing and deploying our own distributed ledger fabric since August 2007. An estimated 3.4 million Euro has been granted through these specific projects and leading directly to this work ([[P2P-Fusion](#)], [[P2P-Next](#)], [[QLectives](#)]).

We thank master student Stijn for his help with writing of this draft.

7. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs [[RFC5226](#)] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

8. Security Considerations

From a security perspective, the usage of novel structures such as Trustchain might lead to new kind of attacks. We consider this risk of less importance for a private and consortium network, where all participants are known to the operator and authentication mechanisms are used to restrict access to the network.

For the public blockchain networks, the usage of Trustchain might lead to new kind of attacks. For instance, an attacker might be able to pollute the chain with refusal to sign attacks to decrease trust. The scope of such attacks and security violations needs to be investigated and is not part of this draft.

9. References

[abrahamself]

Abraham, A., "Self-Sovereign Identity", 2017,
<[https://www.egiz.gv.at/files/download/
Self-Sovereign-Identity-Whitepaper.pdf](https://www.egiz.gv.at/files/download/Self-Sovereign-Identity-Whitepaper.pdf)>.

[apostle2017uber]

Apostle, J., "The Uber data breach has implications for us all.", 2014, <[https://www.ft.com/content/
e2bf6caa-d2cb-11e7-a303-9060cb1e5f44](https://www.ft.com/content/e2bf6caa-d2cb-11e7-a303-9060cb1e5f44)>.

[atzori2016blockchain]

Atzori, M., "Blockchain-based architectures for the internet of things: a survey", 2016,
<[https://papers.ssrn.com/sol3/
papers.cfm?abstract_id=2846810](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2846810)>.

[azouvi2017secure]

Azouvi, S., Al-Bassam, M., and S. Meiklejohn, "Who am i? Secure identity registration on distributed ledgers", 2017, <[https://link.springer.com/
chapter/10.1007/978-3-319-67816-0_21](https://link.springer.com/chapter/10.1007/978-3-319-67816-0_21)>.

[boneh2004secure]

Boneh, D. and X. Boyen, "Secure identity based encryption without random oracles", 2004,
<[http://link.springer.com/content/pdf/10.1007/
b99099.pdf#page=454](http://link.springer.com/content/pdf/10.1007/b99099.pdf#page=454)>.

[brown2017introducing]

Brown, R., "Introducing R3 Corda: A Distributed Ledger for Financial Services", 2016,
<[http://www.r3cev.com/blog/2016/4/4/introducing-r3-corda-
a-distributed-ledger-designed-for-financial-services](http://www.r3cev.com/blog/2016/4/4/introducing-r3-corda-a-distributed-ledger-designed-for-financial-services)>.

[cong2017blockchain]

Cong, K., Ren, Z., and J. Pouwelse, "A Blockchain Consensus Protocol With Horizontal Scalability", 2017,
<[https://repository.tudelft.nl/islandora/object/
uuid:86b2d4d8-642e-4d0f-8fc7-d7a2e331e0e9](https://repository.tudelft.nl/islandora/object/uuid:86b2d4d8-642e-4d0f-8fc7-d7a2e331e0e9)>.

[FIPS180-4]

Gallagher, P., "Secure hash standard (shs)", 2008,
<http://www.cs.haifa.ac.il/~orrd/IntroToCrypto/online/fips180-3_final.pdf>.

[jethanandani2017accounting]

Jethanandani, M., "Accounting in NETCONF and RESTCONF", 2017, <<https://www.ietf.org/id/draft-mahesh-netconf-accounting-03.txt>>.

[johnson2001elliptic]

Johnson, D., Menezes, A., and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)", 2001,
<<http://www.springerlink.com/index/0L7A1W9W38XL6W6R.pdf>>.

[kokkola2011payment]

Kokkola, T., "The payment system: Payments, securities and derivatives, and the role of the Eurosystem.", 2011,
<<https://www.ecb.europa.eu/pub/pdf/other/paymentsystem201009en.pdf>>.

[mcmillan2014inside]

McMillan, R., "The inside story of Mt. Gox, Bitcoin's \$460 million disaster.", 2014,
<<https://www.wired.com/2014/03/bitcoin-exchange/>>.

[miller2016honey]

Miller, A., Xia, Y., Croman, K., Shi, E., and D. Song, "The honey badger of BFT protocols", 2016,
<<http://dl.acm.org/citation.cfm?id=2978399>>.

[nakamoto2008bitcoin]

Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system", 2008, <http://www.academia.edu/download/32413652/BitCoin_P2P_electronic_cash_system.pdf>.

[otte2017trustchain]

Otte, P., de Vos, M., and J. Pouwelse, "TrustChain: A Sybil-resistant scalable blockchain", 2017,
<<https://www.sciencedirect.com/science/article/pii/S0167739X17318988>>.

[P2P-Fusion]

"P2P-Fusion project", 2018,
<http://cordis.europa.eu/project/rcn/105290_en.html>.

[P2P-Next]

"Next Generation Peer-to-Peer Content Delivery Platform", 2018, <http://cordis.europa.eu/project/rcn/85326_en.html>.

[pouwelse2017trustlaws]

Pouwelse, J. and M. de Vos, "Laws for creating trust in the blockchain age", 2017, <https://github.com/blockchain-lab/shared_vision_towards_programmable_economy>.

[QLectives]

"QLectives project", 2018, <http://cordis.europa.eu/project/rcn/89031_en.html>.

[resnick2002trust]

Resnick, P. and R. Zeckhauser, "Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system", 2002, <[http://www.emeraldinsight.com/doi/pdf/10.1016/S0278-0984\(02\)11030-3](http://www.emeraldinsight.com/doi/pdf/10.1016/S0278-0984(02)11030-3)>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [RFC 5226](#), DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

[TUDelft2018trustchain]

Pouwelse, J., "Trustchain - creating trust with software", 2018, <https://play.google.com/store/apps/details?id=nl.tudelft.cs4160.trustchain_android>.

[vukolic2015quest]

Vukolic, M., "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication.", 2011, <http://link.springer.com/chapter/10.1007/978-3-319-39028-4_9>.

[yan2008trust]

Holtmanns, S. and Z. Yan, "Trust modeling and management: from social trust to digital trust.", 2008, <<http://lib.tkk.fi/Diss/2007/isbn9789512291205/article1.pdf>>.

Author's Address

Dr. J.A. Pouwelse (editor)
Delft University of Technology
Delft
Netherlands

Phone: +31 15 2782539

Email: j.a.pouwelse@tudelft.nl