Monoid

towards a ledger for integrity and authenticity of digital identities, decentralised.

PGP-Fingerprint: AB0F AB87 984B A721 BBF3 179E E87B A6D1 2242 C993

October 10, 2017 | v.0.4 (Draft)

Abstract. We propose *Monoid*, a design for a decentralised, authenticated and privacy preserving database for cryptographic keys and identity data. We designed it to meet both possible legal compliance requirements for cryptographic authentication and integrity, as well as the demand for decentralised, trust-minimising key and credential distribution for digital identity.

1 Introduction

> In a sense, communication networks can be defined entirely by who has cryptographic keys.
>
> Whitfield Diffie (Cryptographer)

Most security protocols and analyses of those *assume* trust in a particular entity in their network ontology, which distributes keys for a given entity. In the following, we will propose primitives, frameworks and procedures for the *Monoid*. Our proposal aims to contribute to overcome the trust-problem induced by the premise of third-party-trust. We aim to contribute in high generality, i.e. from a perspective where we consider the following as *ontologically equivalent*:

τ*rust anchors*. Certification authorities (CA's) for X.509 certificates, DNS root servers for TLSA records, PGP-Keyservers are - in a sense - 'trust anchors' in their corresponding cryptographic trust-ontology. Trust is 'rooted' in their integrity, as they are authoritative for some bindings of keys to identities. Usually this role manifests in form of a trusted public-key-infrastructure (PKI) which aims to guarantee two properties for the keys utilised in later applications: *authenticity* and *integrity*. Yet, there is no rigorous reason to trust in 'trusted third parties', i.e. a CA or a key server, besides common sense and belief. In contrast, there is often no mechanism available to audit the key bindings for other parties, i.e. to validate their assumptions regarding *authenticity* and *integrity* of the cryptographic backbone hold. We find these assumptions to be unjustified for two reasons:

i. The security measures undertaken by a trusted third party are usually not publicly available, therefore intransparent. Key servers, CAs or identity providers are black boxes; when using a particular network, (locally) generated key data from one client is signed and transferred to a central entity. Later on, when other clients retrieve a particular key from a server, one has no way to be sure of its integrity.

ii. It is not traceable wether or not a trusted third party is malicious and/or compromised. Quite the opposite is often the case: CAs have economic incentives to keep failures and flaws a secret.

iii. From a network topological viewpoint, any centralised PKI has an inherent single point of failure which is a distinguished target vector for an attack, as a compromised key server is an 'ideal' Man in the Middle. Therefore when protocols utilise a PKI, key servers are a security bottleneck; a protocol can be no more secure as it's corresponding key serving entity.

1.1 RELATED WORK.

We give an overview of recent related proposals, which aims to provide a consistent and secure distribution model for cryptographic key material. While the related systems differ widely, they all aim - more or less - to securely distribute bindings of the form:
$\mathcal{I} \longrightarrow \mathcal{K}$, where $\mathcal{K}$ is cryptographic key material - usually public keys - and $\mathcal{I}$ is an abstract representation of an entity/identity, i.e. an IP- or an Email-Address, a username, pseudonym or a generalisation thereof.

NAME SYSTEMS.

DNSSEC [1] adds some security features to the DNS; while it mitigates some known weaknesses (i.e. cache poisoning and unauthenticated DNS responses), especially the distribution of DANE's TLSA records via the DNS inherit attack-vectors from the authoritative approach the DNS took, such as DoS and the possibility of compromised roots.
NAMEID [2] is based on NAMECOIN's [3] shared high integrity identity ledger, which implements Bitcoin's PoW mechanism. Until today, no successful major attacks on either Bitcoin or it's offspring have been reported. However, Namecoin suffers from similar issues as Bitcoin, i.e. little scalability, wasted resources and the lack of authentication.
The GNU NAME SYSTEM (GNS) [4] implements a $R5n$ [5] hash-table as a decentralised, censorship-resistant ledger to securely store and distribute zone-records. In contrast to the DNS, GNS entities are self-authoritative; Entities may use 'pet names' - i.e. arbitrary key-value bindings - for their name-zones and delegate authority over sub-spaces to other entities they trust. The GNS aims to be able to replace both, the DNS as a name system and as a distribution mechanism for puplic-key material. Additionally, the name resolution mechanism is privacy preserving and encrypted symmetrically, ruling out the possibility to enumerate the name space, which is a privacy issue in the DNS protocol and an inherent problem in Namecoin's public ledger..
However, GNS does not give any guarantees on the integrity of the bindings stored in the distributed $R5n$ hash-table. GNS's trust-ontology is similar to the WoT; Nodes are - in general - unauthenticated, therefore name-spoofing and attacks on the decentralised resolution mechanism - including equivocation - may be possible. Therefore, the GNS is - for now - not compliant to current standard regulation.

OTHER APPROACHES TO ESTABLISH TRUST TO IDENTITY.

CERTIFICATE TRANSPARENCY [6] provides a publicly auditable append-only log for X.509 TLS certificates. Information stored in this log is distributed to clients by monitors, ran by certification authorities. Therefore, there is no abstract entity added to its trust ontology, as the log servers are controlled by the certificate authorities themselves.

NICKNYMs [7] approach is a combination of TOFU and X.509, together with notary servers (CAs), forming a federated web-of-trust (WoT) with a non-equivocation mechanism that users can query. Besides internal failures or compromised notaries, it is - for today - prone to enumeration attacks and DoS on Nym-servers.

KEYBASE [8] introduces social validation via social media accounts. Users publish statements about their key bindings through different social media channels; if later other users want to gather evidence for the correctness of these bindings, they can query a server, that answers with the 'site's global state'. From this state, the querying user learns the location of the resources she needs, in order to gather evidence for some other nodes key-binding, which is in question. Thus, a compromised Keybase server can potentially fork the site's state and equivocate about a key binding.

UPORT [9] aims to introduce secure identities via identifiers - addresses of so called proxy 'smart contracts' - stored in the Ethereum blockchain, able to interact with other 'smart contracts'. Additionally, Uport aims to use IPFS as an off-chain registry for identity data. Identities are able to act as a self-sovereign certificate authority, introducing similar attack vectors to Uport as are applicable to Namecoin, caused by its lack of authentication and accountability (i.e. name spoofing and fake ID's, lack of legal compliance).

U-PROVE [10] proposed a cryptographic token, generalising and strengthening the concept of a cryptographic certificates. Ontologically, U-prove knows three different roles; Issuers, provers and verifiers. Issuers have the same role as certification authorities - being the trust anchor of the tokens they issue - while provers

are common users, that intend to generate a credential to prove certain claims about their identity towards a verifier (i.e. an online service, etc.). While U-prove guarantees certain cryptographic properties, such as untraceability, unlinkability and selective disclosure of tokens, it assumes secure distribution of issuer's public keys (as it is done today with CA root-keys) and is not concerned with issuers equivocating about user identities towards other users or verifiers.

CONIKS [11] retains the notion 'of service providers issuing authoritative name-to-key bindings within their namespaces' (CA's) as well, but extends the trust ontology in the following way; users are enabled to verify consistency of their key bindings with a proof of consistency - extending the function of certificates - which were distributed by identity providers. However, Coniks assumes, just as U-prove, a separate PKI for identity providers keys, leading into some security regress, i.e. the hen-or-egg problem. Central identity providers create a decided attack vector: From inside, via authoritarian decisions or compromises, as well as DoS. While Coniks enables users to detect the former at a later point in time, it is not able to 'deter a service provider from attacking its users openly'.

CLAIMCHAIN [12] is a functional abstraction of standard certificates, similar to U-prove tokens, but with a more user centric network-ontology, especially tailored to deliver privacy preservation to an identity system. The structure enables general claims to be made by users about themselves or each other, in particular about their respective key material. Claimchain's verification mechanism is not specified, it can therefore be deployed with different schemes, ranging from traditional CA's to WoT. Its security properties are highly influenced by this particular choice.

SOVRIN [13] has proposed a 'public permissioned ledger' for user identity data. It follows the idea of a federated WoT, operated and administrated by the Sovrin Foundation. It aims to deploy a blockchain derivative that is permissioned, i.e. there's a decided amount of so called 'trustees' and 'stewards', that reach consensus on the state of the ledger, following a modified version of the RBFT [14] protocol. Claims on identities are stored with tokens as described by U-PROVE, where 'stewards' correspond to 'issuers' in Uprove's diction. Due

to its semi-federated organisational structure, it has similarities to the DNS, and therefore could - if deployed at a large scale - raise similar tech-political questions.

1.2 OVERVIEW.

CONTRIBUTION.

Our proposition is well understood as a collage; we arrange several proposals from past years to synthesise[1] a general framework for a decentralised identity ledger - the *Monoid* - tailored to introduce strong integrity and authenticity to digital identity in a decentralised manner. Especially, we aim to tie together some loose ends, i.e. we bridge approaches taken by the 'blockchain community', regulatory demands, together with recent proposals from the scientific computer-science and cryptography discourse.

DESIGN GOALS.

i. *Integrity*. | Equivocation about identities is hard, for both, external adversaries as well as for the entity controlling the identity.
ii. *Trust – minimising*. | Discovery of other's identities is easy and requires a minimal amount of trust in other parties.
iii. *Checks and balances*. | Entities are enabled to proactively shape their identity, contribute in the systems trustworthiness and repel attacks with ease.
iv. *Privacy*. | Interaction does not leak any unnecessary information. Social graphs are unobservable for eavesdroppers.

1 Philosophically, we follow G.W. Hegel's principle of *Dialectic Aufhebung*, to propose a framework for identity authentication and integrity protection, puzzling together different notions.

Approach.

We now give an overview of our architecture and fix key features of our proposal. With *Monoid*, we follow the concept of a distributed ledger for cryptographic identities, enabling integrity protection *and* secure authentication, both in a decentralised manner.
In general, the *Monoid* is a distributed Merkle tree. As a distributed ledger, it has similarities to blockchains and distributed hash tables. In contrast to the usual blockchain procedure, we describe a protocol for consensus we call 'partial weighted proof of identity' (PWPOI), a reputation-system-based consensus mechanism, with the goal to adapt to experiences that have been made with recent constructions for decentralised ledgers:
a. Scalability: The PoW consensus mechanism does not adapt to a large number of users easily. As an increased number of users does usually not come with a proportional increase in the systems resources, PoW converges towards centralisation, besides other downsides.
b. If write privileges are granted to easily, integrity suffers. This has happened to early peer to peer networks, i.e. bit torrent. This could as well happen to Namecoin, as arbitrary names may be registered without authentication.
Our PWPOI mechanism aims to solve these two problems, i.e. it is a scalable method for decentralised consensus, tailored to use identity authentication as a 'stake' for an identity system. It is 'partial', in the sense that nodes are assigned parts of the *Monoid* to watch and gossip on, depending on the resources they have access to. We define a metric, that may be used to compute 'weights' corresponding to the amount of influence a given nodes 'gossip' on the *Monoid's* (-partial) state has on other nodes perspectives of the *Monoid's* state. Nodes then interact in a gossiping protocol to gather evidence in order to determine the *Monoid*'s state.

Scope and limitations:

We aim to describe the macroscopy of a shared ledger, the *Monoid*, and corresponding mechanisms. We assume that the role of a privacy preserving identity data structure (PPIS) is already filled by a data structure such as described in Claimchain or Uprove
Monoid's role in the security pipeline is that of a decentralised high-integrity key-value store, where keys are cryptographic public keys and values are hash digests, corresponding to and derived from the PPIS of certain entities.
Additionally, the PWPOI consensus mechanism is based on multiple authentications and therefore also contributes a explication of the 'social verification mechanism', Claimchain did not specify so far.
However, while we give mathematical definitions and formalisations to some extend, our proposal will often remain abstract; i.e. it falls short to give details, especially with respect to implementation.

2 PRELIMINARIES.

2.1 NOTATION.

Hashfunctions. With $\eta(X)$ we denote the cryptographic hash of data X.

Encryption. With $\mathcal{E}_{\epsilon_i}(X)$ we denote the cipher text of plaintext X with respect to a cipher $\mathcal{E}$ and a key ϵ_i.

Decryption. With $\mathcal{D}_{\delta_i}(X)$ we denote the decipherment of $\mathcal{E}_{\epsilon_i}(X)$. Note that ϵ_i and δ_i do not differ necessarily.

Signatures. As a signature $\mathcal{S}_{\epsilon_i}(X)$, we denote the tuple $(\mathcal{E}_{\epsilon_i}(\eta(X)), \mathcal{D}, \delta_i, X)$.

2.2 ASSUMPTIONS.

$\alpha)$ *random oracle model* : η is pre-image resistant and collision resistant.

$\beta)$ *forgery resistance* : $\phi(\mathcal{E}_{\epsilon_i}(X)) = X$ for some functon ϕ if and only if $\phi = \mathcal{D}_{\delta_i}$

$\gamma)$ *authentic entities*. In a network, the majority of 'authenticated' entities does not affect the network adversely.

2.3 DEFINITION AND PROPOSITION.

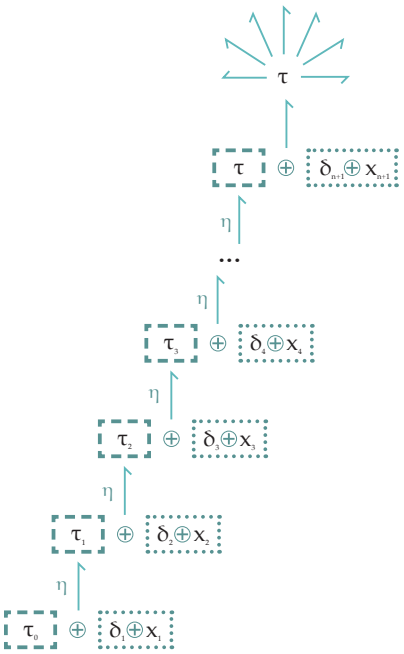

Fig. 2.1: Illustration of a τ-tree

Definition 1. Let $(\delta_i, \mathcal{X}_i)_{i\in\mathbb{N}}$ be a series of pairwise distinct public keys and corresponding (identity-) data. A τ-tree τ is a recursive series $(\tau_i)_{i\in\mathbb{N}}$ of hash values defined as follows:

$$\tau_0 = \eta(\delta_0 \| \mathcal{X}_0)$$
$$\tau_{i+1} = \eta(\tau_i \| \delta_i \| \mathcal{X}_i)$$

Proposition 2. *τ is a directed acyclic Merkle-Graph*

Proof. Let $V = \{\tau_i \mid i \in \mathbb{N}\}$ be the set of vertices of τ. The set of edges is then given by all ordered pairs $E = \{\{\tau_i, \eta(\tau_i \| \delta_i \| \mathcal{X}_i\} \mid i \in \mathbb{N}\} = \{\{\tau_i, \tau_{i+1}\} \mid i \in \mathbb{N}\}$. Assume $\exists\ E_c \subseteq E,\ s.t.\ E_c = \{\{\tau_j, \tau_{j+1},\}, \{\tau_{j+1}\, \tau_{j+2}\}, \ldots, \{\tau_{j+k}, \tau_{j+1},\} \mid j \in \mathcal{J} \subset \mathbb{N}\}$. This gives us: $\eta(\tau_j \| \delta_i \| \mathcal{X}_j = \eta(\tau_{j+k} \| \delta_{j+k} \| \mathcal{X}_{j+k})$ in contradiction to assumption α). Therefore τ is acyclic.

□

Remark. This short proof of τ being acyclic implies, that if a series of τ-trees $(\tau_i)_{i\in\mathcal{I}}$ all have current final hash value τ_f, then $\tau_i = \tau_j,\ \forall i, j \in \mathcal{I}$. The converse is true as well. This observation gives us:
$(\delta_i, \mathcal{X}_i)_{i\in\mathbb{N}}$ *consistent* $\leftrightarrow$ τ_f *consistent*.

Definition 3. Let $(\tau_i)_{i=1,\dots,n}|\ n\ even$, be a series of pairwise distinct τ-trees with final hash $\tau_{f,i}$. Then $\mathcal{M}_{j,k}$ depicts entries of order j and position k in the Merkle-tree $\mathcal{M}$ generated from $\tau_{f,i}$. We call $\mathcal{M}$ a *Monoid*[1], which is defined as follows:

$$\begin{aligned}
\mathcal{M}_{0,k} =& \eta(\tau_{f,0}||\tau_{f,1}), \eta(\tau_{f,2}||\tau_{f,3}), \dots && , \eta(\tau_{f,n-3}||\tau_{f,n-2}), \eta(\tau_{f,n-1}||\tau_{f,n}) \\
\mathcal{M}_{1,k} =& \quad \eta(\mathcal{M}_{0,0}||\mathcal{M}_{0,1}), \dots && , \eta(\mathcal{M}_{0,\frac{n}{2}-1}||\mathcal{M}_{0,\frac{n}{2}}) \\
& \quad \vdots && \vdots \\
\mathcal{M}_{m-1,k} =& \quad \eta(\mathcal{M}_{m-2,1}||\mathcal{M}_{m-2,2}), \eta(\mathcal{M}_{m-2,3}||\mathcal{M}_{m-2,4)} \\
\mathcal{M}_{m} =& \quad \eta(\mathcal{M}_{m-1,1}||\mathcal{M}_{m-1,2})
\end{aligned}$$

Remark. $\mathcal{M}$ is a slightly altered Merkle-tree. Therefore, its immutability can be reduced to the integrity protection of $\mathcal{M}_m$, which can be shown analogue to the proof of Proposition 2.

1 **not** in the algebraic sense, as hashing is non-associative

3 CORE: AUTHENTICITY | INTEGRITY

The definition of τ-trees in the last chapter was intended to create an immutable distributed data base to store values, representing identities (i.e. claimchains or U-prove tokens) together with their corresponding public keys.
We gave a proof, that immutability and - as a result - integrity of the stored data is reducible to integrity of one hash-value. We therefore envision the following general construction for the *Monoid*, based on recursive hashing:

$$\underbrace{chains}_{\text{claimchain, u-prove token}} \xrightarrow{\eta} \underbrace{\tau - trees}_{\text{Def. 1}} \xrightarrow{\eta} \underbrace{Monoid}_{\text{Def. 3}}$$

3.1 INTEGRITY.

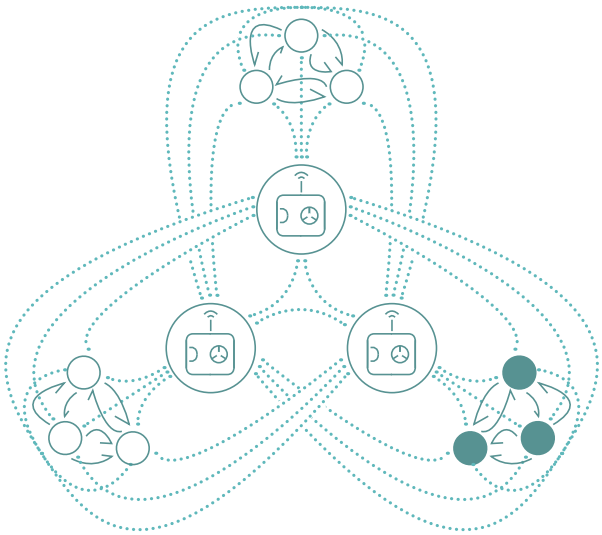

Fig. 3.1: The distribution of the *Monoid*. Only nodes of 0th order and nodes of 1st order depicted, while there may be nodes of higher order in general.

Next, we describe a partitioned and decentralised network to distribute the *Monoid*, giving participating entities access to the keys. The proposed design is depicted in the figure on the left. We tailored it to approach the following design goals:

i. Equivocation about identities is hard, for both, external adversaries as well as for the entity controlling the identity.
ii. Discovery of other identities is easy and requires a minimal amount of trust in other parties.

Abstractly, the τ-tree data structure - if suitably distributed - would be sufficient to reach these design goals. This approach was taken with blockchain-designs, yielding their scalability issue. In order to overcome this obstacle, we decided to take a partitioning approach, i.e. to distribute the *Monoid* data structure partially. We only give a sketch of the partitioning.
In general, nodes are ordered by the amount of resources at their disposal and the size of the partition of the *Monoid* they are able to process: For convenience,

we consider only two types of nodes in two layers, *populars* and *ladders*[1] while, in general, additional layers may be introduced to increase the fragmentation if necessary.

populars | 0*th order*. Popular nodes are both subscribers and publishers of the current state of a particular subset of trees τ_i, representing consistently stored data in a subset of the network. Nodes are able to audit the *Monoid* by checking the integrity of the key-value bindings in their partition by verifying τ_i. They further periodically sign and publish the current τ-values of their partition of the *Monoid* to other nodes. Therefore, every peer has a maximal amount of different and independent sources for both the τ-trees and the corresponding key and identity data.

ladders | 1*st order*. If machines have enough resources to process the data generated by all subnetworks, they may become a ladder node. They live in the intersection of all subnetworks and therefore operate as a *popular* node in *any* subnetwork. They additionally use the same procedures as nodes of lower order to compute the *Monoid* from τ-trees and negotiate consensus among each other as well.

Remark. Both layers are unable to equivocate about the *Monoid*'s state, as they mutually audit each other. If additional layers were introduced in between, this would further strengthen the auditing mechanism.

1 The diction 'ladder node' is derived from a notion of L. Wittgenstein in the following sense: Centralised nodes fulfil the purpose of a ladder and may become unnecessary at a later point in time, 'the ladder may become exposable after climbing with it'. I.e. we favour decentralisation but utilise central nodes for authentication - mainly for legal and governmental reasons -, but as well to increase scalability and availability.

3.2 AUTHENTICITY.

> A person is a person through other persons.
>
> Proverb, Ubuntu philosophy [15]

The draft we described so far, is prone to sybil attacks and name spoofing. Our approach against this kind of attack is identity authentication, i.e. certificates from 'trusted' parties of some kind. As stated in premise γ, we assume that if nodes are controlled by *authenticated* entities, a majority of them will not affect the network adversely. Further, authentication is for many applications of digital identity required by law (i.e. digital signatures in e-gov).Authentication to just *one* party is the theoretical minimum required to have any authentication in a system at all.
Usually, profound ways of authenticating a party's identity rely on tokens and certificates issued by (governmental-) trusted third parties, serving as trust-anchors (i.e. passports). Our architecture entails this model, but in contrast, we envision to invert *the burden of proof* in the following sense: authentication of new nodes with a *ladder node* is a requirement in order to write an identity to the *Monoid*. Yet, we depict nodes in the ladder layer as *mistrusted third party*, i.e. this authentication is to be verified in order to accumulate trust by other (popular-) nodes. We will further describe this notion of authentication, model it in a 'mistrust metric' and the PWPOI mechanism for consensus in the following sections. We aimed to reach the following design goal:

iii. Checks and balances. | Entities are enabled to proactively shape their identity, contribute in the systems trustworthiness and repel attacks with ease.

3.2.1 MISTRUSTED THIRD PARTIES | BOUNTY HUNT

We give a sketch of an authentication framework, that entails the current trusted third party model. Generally speaking, we invert the *the burden of proof*: Authentication (signatures on key-value bindings stored in the *Monoid*) issued by third parties (the ladder nodes in our diction) are *accepted* and potentially legally binding, but not *trusted*. The *mistrusted third parties* are now assigned the task to enhance the trust in the validity of their authentication. In order to do so, they start a *bounty hunt*. This yields a procedure for authentication[2], that synthesizes and incentivizes two common authenticity models: *certification authorities* and the *web of trust*.

i) An entity N gets authenticated via a certificate, worth a bounty $\mathcal{B}$, by a ladder node $\mathcal{L}$. For this authentication $\mathcal{L}$ receives 50% of the bounty $\mathcal{B}$.

ii) In the following, the remaining 50% of $\mathcal{B}$ get utilized as an incentive for peers to iteratively re-authenticate N and audit the authentication of $\mathcal{L}$. $\mathcal{B}$ gets distributed in a slowly converging series among k *bailers*, for instance as geometric series $\mathcal{B}_k = \frac{\mathcal{B}}{2^k}$, s.t. $\sum_{i=1}^{k} \mathcal{B}_k \xrightarrow{k\to\infty} \mathcal{B}$.

Remark. In theory, this procedure converges towards *optimal authentication* (i.e. full mutual authentication with every other node) and only terminates for an infinite number of verifiers, but is practically bound by $n-1$ possible authentications and - even more so - by the fast decreasing incentive for new *bailers*.

2 While the analogue measures for authentication are crucial, they are outside of our scope.

3.2.2 AUTHENTICATION-GRAPH | AUTHENTICATION-METRIC.

We now specify the model and terminology of *Monoid*'s mechanisms for *authentication* and *consensus*, assuming authentication occurs within the framework we sketched. In general, our approach is to link authentication to consensus, according to premise γ. We specify a non-binary reputation metric in order to compute weights for a nodes gossip on the current state on (a partition of) the *Monoid*.

heuristics. We define a metric μ, that aims to model the following heuristics with regard to mistrust of authentication among a set of entities:
i) Mistrust increases, as a path of trust grows longer from one entity to another.
ii) Mistrust decreases with the amount of mutually familiar entities on a path.
iii) Mistrust decreases, with the amount of *disjoint* paths of trust from one entity to another.

These heuristics motivate the following definitions:

Definition 4. *authentication graphs.*
Consider a set of nodes $\mathcal{N}$, $i = 1, ..., k$. Let there be an edge $e = (x_i, x_j)_{x_i \neq x_j} \in \mathcal{E}$ between two nodes iff x_i and x_j are mutually authenticated. This defines an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Definition 5. *authenticated networks.*
Consider an edge $e = (x, y)_{x \neq y} \in \mathcal{E}$ and the closed neighbourhood of a node $\mathcal{N}_{\mathcal{G}}[x]$. We define:

$$\mu' : \mathcal{E} \to [0, 1],$$
$$(x, y) \to \frac{2}{|\mathcal{N}_{\mathcal{G}}[x] \cap \mathcal{N}_{\mathcal{G}}[y]|}$$

The triple $(\mathcal{V}, \mathcal{E}, \mu')$ is a *weighted graph* or a *network*. We now enrich μ' to receive a metric on $\mathcal{G}$.

Definition 6. *mistrust – metric.*
Let be $\Pi_{x,y}$ be the set of all *paths* from x to y in $\mathcal{G}$. If $\Pi_{x,y} \neq \emptyset$, let be

$$\pi^s_{x,y} := \underset{\pi_{x,y} \in \Pi_{x,y}}{\operatorname{argmin}} \sum_{e \in \pi_{x,y}} \mu'(e),$$

i.e. a shortest path with highest degrees of authentication from x to y. We define $\mu : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ to receive a metric as follows:

$$\mu(x,y) = \begin{cases} 0 & \text{, if } x = y \\ \frac{|\pi^s_{x,y}| \left(\sum_{e \in \pi^s_{x,y}} \mu'(e) \right)}{|\Pi_{x,y}|} & \text{, if } x \neq y \end{cases}$$

Proof. Non-negativity follows, as only positives are involved. Identity follows by definition.
For symmetry consider: $x, y \in V$ with $\Pi_{x,y} \neq \emptyset$. We get $\Pi_{x,y} = \Pi_{y,x}$ and therefore $\pi^s_{x,y} = \underset{\pi_{x,y} \in \Pi_{x,y}}{\operatorname{argmin}} \sum_{e \in \pi_{x,y}} \mu'(e) = \underset{\pi_{y,x} \in \Pi_{y,x}}{\operatorname{argmin}} \sum_{e \in \pi_{y,x}} \mu'(e) = \pi^s_{y,x}$

Let be $x, y, z \in \mathcal{V}$. We have

$$\begin{aligned} \mu(x,z) = \frac{|\pi^s_{x,z}| \left(\sum_{e \in \pi^s_{x,z}} \mu'(e) \right)}{|\Pi_{x,z}|} &\leq \frac{|\pi^s_{x,y}| \left(\sum_{e \in \pi^s_{x,y}} \mu'(e) \right)}{|\Pi_{x,y}|} + \frac{|\pi^s_{y,z}| \left(\sum_{e \in \pi^s_{y,z}} \mu'(e) \right)}{|\Pi_{y,z}|} \\ &= \mu(x,y) + \mu(y,z), \end{aligned}$$

as

$$|\pi^s_{x,z}| \left(\sum_{e \in \pi^s_{x,z}} \mu'(e) \right) \leq |\pi^s_{x,y}| \left(\sum_{e \in \pi^s_{x,y}} \mu'(e) \right) + |\pi^s_{y,z}| \left(\sum_{e \in \pi^s_{y,z}} \mu'(e) \right) \text{ and}$$

$$\Pi_{x,z} \subset (\Pi_{x,y} \cap \Pi_{y,z}).$$

Thus, $(\mathcal{V}, \mathcal{E}, \mu)$ is a metric space. □

3.3 CONSENSUS | ACCOUNTABLE GOSSIPING.

For consensus, we envision a combination of an epidemic algorithm [16], a forward-secure, identity-based multi-sign scheme (IBMS) as in [17] together with the mistrust-metric μ we defined in the last section.
We depict it as *accountable gossiping* or *partial weighted proof of identity* (PW-POI). Entities $\mathcal{N}_i$ with access to the *Monoid* periodically sign the current τ values they are auditing and publish it to other nodes. A gossip on the state of a τ-tree is then multi signature of the form $\mathcal{S}_{\epsilon_1,\ldots,\epsilon_n}(\tau_k)$.

INTRA-PARTITIONAL CONSENSUS.

Nodes, that receive gossip from other nodes, which are auditing *the same* partition of the *Monoid*, have all information at their disposal, that is necessary to check the validity of the signatures.

i) They periodically compute τ_k for the state of their partition and compare it to other gossip they receive.

ii) If no inconsistencies are detected, they accept the gossip on the current state of τ_k, use the IBMS to sign what they received in the previous step with their own private key and republish the result.

iii) Else, they reinitialise the procedure with their current τ value and publish a concurrent gossip.

iv) Eventually, after a time to be specified, some hash τ_k receives the majority of signatures and is considered trustworthy.

INTER-PARTITIONAL CONSENSUS.

Nodes, that receive gossip from other nodes, which are auditing a $different$ partition of the $Monoid$, do not have all information at hand, that is necessary to check the validity of the signatures. However, there's is at least *one* authentication-path to every other node, due to the obligatory authentication with a ladder node. We further assume, that they have the ability to compute the *authentication network* (Definition 5). Nodes are then able to gather evidence regarding the state of the $Monoid$ outside of their partition as follows:

i) If a node receives gossip from other nodes $outside$[3] of their partition, it computes its own mistrust towards the publishing node as in Definition 6.

ii) Optionally, a node has the ability to compare the gossip published by $popular\ nodes$ with the gossip of $ladder\ nodes$, if she chooses to prioritise them as trusted sources.

iii) In any case, a node $\mathcal{X}$ chooses to trust the state of the $Monoid$ it has most evidence for,[4] with respect to the $mistrust - metric$ and its choice in ii).

3 if the publisher and the subscriber audit partitions of the $Monoid$ with non-empty intersection, the receiving node first verifies the correctness of they key-value bindings in the intersection. If it detects an inconsistency, the gossip is to be disposed.

4 i.e. the gossip $\mathcal{S}_{\epsilon_1,...,\epsilon_n}(\tau_k)$ from nodes $x_1, ..., x_k$ where $\sum_{i=1}^{k} \mu(\mathcal{X}, x_i)$ is minimal.

4 SUMMARY.

4.1 CONCLUSION.

After briefly sketching our trust-ontology and specifying our assumptions, we have described the *Monoid*, a decentralised ledger for digital identities with strong integrity and authenticity protection. We gave definitions for all required building blocks which are necessary to construct the *Monoid* ledger and sketched a layered decentralised architecture to distribute it. We further gave high-level descriptions of procedures for integrity, authentication, to establish trust and consensus. However, we did not give any specifics with respect to implementation.

4.2 FUTURE WORK.

We aimed to make our premises transparent and give definitions and proofs with some rigour. However, our proposal fell short of several aspects we want to address in the future:
First, we assumed that the majority of *authentic entities* in a network does not affect the network adversely (γ). This assumption is speculative and hard to model rigorously. It is based on the notion of *authenticity*, which is to our knowledge, an unanswered question more of philosophical nature. A future line of argument

should therefore be an empirical examination of the validity of this premise.
Before implementational choices are made and evaluated, we will contribute more theoretical work, i.e. define an adversary model, state security propositions and proofs thereof.
Further, we described the distribution and the auditing of the *Monoid*, which require partitioning it for scalability. Yet, while the partitioning mechanism may have severe implications - including but not limited to security - we did not give any specifics on how partitioning should happen. A next step therefore is implementing the *Monoid* as a distributed hash table, i.e. with *Kademlia* [18] or more advanced distributed-hash-table algorithms such as *R5n* [5].
The particular implementation also impacts another problem, that our draft assumed to be solvable; access and computability of authentication graph. In our proposal we assumed a 'birds eye view' for every node, i.e. full knowledge of the authentication graph and computational feasibility of the algorithms necessary to compute the values of the mistrust-metric μ. An implementation - for instance with A* - could indicate, if design changes are necessary.
Another open aspect is how in particular authentication of new nodes should take place; there are many different approaches to this problem and it is an open research questions, ranging from manual hash-fingerprint comparison to whole scientific fields (i.e. biometrics). We do not aim to specify the particular measures undertaken for authentication; while they are critical, they are interchangeable in our model.

5 APPENDIX: ON THE RELATION TO BLOCKCHAINS.

The ledger we proposed has similarities to the original Bitcoin-Blockchain [19], but favours a *one vote per node* over *one vote per CPU* paradigm for consensus. This yields a trade-off; it allows sybil attacks, where a large group of adversary machines compromises the consensus mechanism by outvoting. This issue is tackled by our setup in several ways: First, the network is partitioned and replicated as described earlier. Second, our design works with authentication, a cryptographic property that was later forced into blockchain environments (i.e. today, before able to buy Bitcoins, users are often legally obliged to analogously authenticate to a third party). We opt out from Bitcoins pseudonymity and introduce authentication. Although this is a drawback regarding Zooko's Triangle, we consider this the best option, as todays laws usually require a certain amount of identity validation before intensive application is permitted. Our design therefore tries to find a reasonable compromise between the central authentication required by law and as most decentralisation and privacy as possible for an identity system to be deployable on a large scale.

Yet, the biggest advantage of favouring the *one vote per node* paradigm is in our opinion, that it removes the necessity of mining new blocks via proof of work, which is an intentional and inherent blockchain inefficiency that increases over time together with the difficulty of the proof of work. We find a few inherent

drawbacks that mining brings with it; first, it is an obvious and intentional waste of CPU power, that could otherwise be utilised elsewhere. It is also a waste of energy and therefore of environmental resources, which isn't and may never will be a good foundation to build the backbone of modern infrastructure on. Last, many blockchain based networks centralise themselves over time; as mining becomes more and more difficult, the incentive for individual and independent mining decreases. Miners collaborate to have better chances to get rewarded, while it is out of the reach of the protocol to regulate who controls the mining pool. We sketched a decentralised incentive, that strengthens the systems trust while actually contributing useful work: Authentication.

Bibliography

[1] G. Ateniese and S. Mangard, "A new approach to DNS security (DNSSEC)," in *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.*, pp. 86–95, 2001.

[2] "Nameid." https://github.com/namecoin/nameid.

[3] "Namecoin - A trust anchor for the Internet, url = http://www.win.tue.nl/hashclash/rogue-ca/,."

[4] M. Wachs, M. Schanzenbach, and C. Grothoff, *A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System*, pp. 127–142. Cham: Springer International Publishing, 2014.

[5] N. S. Evans and C. Grothoff, "R^{5}n: Randomized recursive routing for restricted-route networks," in *5th International Conference on Network and System Security, NSS 2011, Milan, Italy, September 6-8, 2011*, pp. 316–321, 2011.

[6] "Certificate transparency." https://www.certificate-transparency.org/, last accessed: 10/17.

[7] "Nicknym." https://leap.se/nicknym, last accessed: 10/17.

[8] "Keybase." https://keybase.io/, last accessed: 10/17.

[9] "UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY." https://whitepaper.uport.me, last accessed: 10/17.

[10] C. Paquin, "U-prove technology overview v1.1 (revision 2)," April 2013.

[11] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "CONIKS: bringing key transparency to end users," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pp. 383–398, 2015.

[12] B. Kulynych, M. Isaakidis, C. Troncoso, and G. Danezis, "Claimchain: Decentralized public key infrastructure," *CoRR*, vol. abs/1707.06279, 2017.

[13] "The Technical Foundations of Sovrin." https://sovrin.org/wp-content/uploads/2017/04/The-Technical-Foundations-of-Sovrin.pdf, last accessed: 10/17.

[14] P. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: redundant byzantine fault tolerance," in *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA*, pp. 297–306, 2013.

[15] M. Eze, *Intellectual History in Contemporary South Africa* -. Palgrave Macmillan US, 2010.

[16] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic Algorithms for Replicated Database Maintenance," *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC)1*, pp. 1—-12, 1987.

[17] M. Bellare and G. Neven, "Identity-Based Multi-signatures from RSA," in *Topics in Cryptology âĂŞ CT-RSA 2007* (M. Abe, ed.), pp. 145–162, 1973.

[18] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, (London, UK, UK), pp. 53–65, Springer-Verlag, 2002.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," May 2009.