Design of NEO ID

2019-06-14

Introduction

Identities are fundamental attributes of humans and they determine the way experience the world. Identities help us evaluate whether to trust another person and they can enable or restrict our access to certain resources or locations. With the growth in internet interconnectivity and digitalization of services, we are now able to interact with thousands of companies and millions of users instantly and in innovative ways. But each time we carry out vital communication in any distributed computer system, we face an inherent risk. This risk arises because we can never be completely certain about the trustworthiness of entities that mediate our online interactions. Some criminals falsify their identity to defraud loans ; also impersonate someone to commit fraud and other crimes by false Facebook account. To eliminate or reduce this phenomenon, we need to be responsible for our digital identities now more than ever. This identity data is usually kept far away from our view and control, in centralized corporate data siloes that are highly-susceptible targets of attacks, negligence or even malicious data selling. The repeatedly-demonstrated failures of these corporate entities affect our social, professional and financial lives with little we can do about it.

Digital identity means condensing real identity information into a digital code. It's a public key that can be queried and identified through the network, related devices, etc. The development phase of digital identity has gone through four phases: centralized identity, federated identity, user-centric identity, and self-sovereign identity. The identifier of the era of centralized identity is the individual's email account, and the back of that represents a real individual. Federated identity can be compared to Facebook, Twitter and Instagram cross-platform login. User-centric identity and self-sovereign identity are closely related and inseparable.Self-sovereign identity is built on the blockchain, which combines the decentralization, distributed system, consensus protocol, hash encryption and other characteristics of the blockchain. So it is on the level of autonomy, security and controllability.

Decentralized identity systems would provide no centralized target for attacks and are more resistant to censorship or malicious intent. Self-sovereign identity systems seek to put the control of a user's identity data back in their own hands,

instead of fragmented across the internet and data centers, sometimes without their knowledge. NEO ID seeks to present such a W3C-compliant solution built on top of the NEO blockchain that returns data sovereignty back to the user.

Definition

"Entity" refers to individuals, legal entities (organizations, enterprises, institutions, etc.), objects (mobile phones, automobiles, IoT devices, etc.) in the real world, and "identity" refers to the entity's identity within the network [1]. We use NEO ID to identify and manage the entities' identities in the network.

In more detail, a NEO ID is a complete digital representation of a person (or app, organization, device, or bot) that is able to make statements about who they are when interacting with smart contracts and other identities, either on-chain or off-chain. This ability to make statements about themselves, without relying on centralized identity providers, is what make NEO ID a system for self-sovereign and distributed identity. Besides, since the centralized trust relationship is simply expressed as trust or distrust in the past, we can hardly quantify how much does Alice trust Bob, so we introduced a rating mechanism for NEO ID which can transforms trust in centralized authority into trust in distributed mathematical calculations.

In the NEO ID system, each entity may simultaneously possess multiple identity documents that correspond to different identities (e.g. academic, employment, personal) and to different roles in the NEO ID system (e.g. trustor, recommender/CA, trustee). These identity documents can issue subjective claims to each other and build dynamic trust graphs used to evaluate the trust between one identity document and another. The formation of trust graphs make relationship become "context-dependent" which means trust can be passed among entities

As a decentralized identity identification protocol, Each NEO ID corresponds to an NEO ID description object in complete anonymity, which is used to record attribute information such as the controller public key of the NEO ID. The description object is exposed as information to be stored in a distributed ledger. For the sake of privacy protection, the description object does not contain any information related to the real identity of the entity by default.

As for the most important thing, a global solution for digital identity must enable every person and organization to verify and safely share highly private information—banking records, tax records, health records. Protecting the privacy of such records is crucial—in some cases even a matter of life and death. So at the heart of NEO ID architecture are three fundamental principles: secure, robust and sustainable, the opportunity to apply the principles in an identity system that protects not just the citizens of one country, or the customers of

[1] https://github.com/ontio/Documentation/blob/master/Ontology-technology-white-paper-EN.pdf

one company, or the members of one social network, but every person and organization in the world who opts to use NEO ID as a global public utility.

Solution

This section describes the solution of NEO ID.

Terminology

This section introduces the primary terms used in NEO ID with well-defined terminology.

entity a thing with distinct and independent existence.[2] or a resource of any kind that can be uniquely and independently identified.[3]

identifier name that identifies an *identity document*. An *identifier* may be a word, number, letter, symbol, or any combination of those.[4]

public key the half of an asymmetric cryptographic key pair designed to be shared with other parties in order to decrypt or verify encrypted communications. In digital signature schemes, a public key is also called a verification key. Public-key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of keys: public keys which may be disseminated widely, and private keys which are known only to the owner.[5] The generation of such keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions. Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.[6]

proposal an expression to describe the set of attribute of the entity or the authorization limit scope of the entity. We will discuss the NEO ID *proposal* in the proposal model section.

metadata data that provides information about other data.[7]

identity document digital document which may be used to prove an entity's identity which contains a proposal of the entity.[8] [9] An *identity document* consists of an unique *identifier*, a *public key* that owned by the entity, a *proposal* associated to the entity and a *metadata*. An *identity document*

[2] https://en.oxforddictionaries.com/definition/entity
[3] Sovrin Provisional Trust Framework, Sovrin Board of Trustees, 28 June 2017
[4] https://en.wikipedia.org/wiki/Identifier
[5] https://en.wikipedia.org/wiki/Public-key_cryptography
[6] Stallings, William (3 May 1990). Cryptography and Network Security: Principles and Practice. Prentice Hall. p. 165. ISBN 9780138690175.
[7] https://en.wikipedia.org/wiki/Metadata
[8] https://en.wikipedia.org/wiki/Identity_document
[9] https://en.wikipedia.org/wiki/Authorization_certificate

can be signed or unsigned. It can be signed with a signature of all of the infomation above.

$$identitydocument = \langle identifier, publickey, proposal, metadata \rangle$$

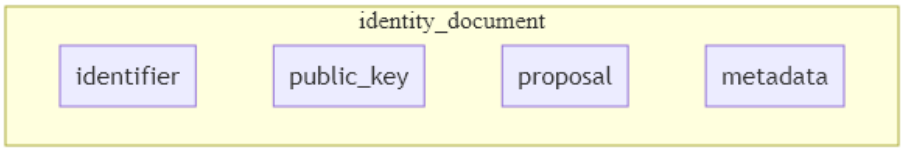


Figure 1: The Structure of Identity Document

rating a value coupled with assertions which means an indication of how much the party generating the Response Set agrees with the assertion being made.[10] In this work, we have chosen to represent *rating* as a continuous variable over a specific range $[-1, +1]$.[11] We discuss the benefits and drawbacks of such an approach in the analysis section. In NEO ID, the *rating* can also be regarded as a value to measure trust or authorization as an alternative to trust.[12]

claim a directed connection that describes a *rating* to an *identity document* issued by another *identity document*'s holder. A *claim* consists of two *identifier*s of *identity document*s, a *rating* and a *metadata*. A *claim* is valid only if the proposal of the to *identity document* implies the proposal of the from *identity document*. An *claim* can be signed or unsigned. It can be signed with a signature of all of the infomation above.

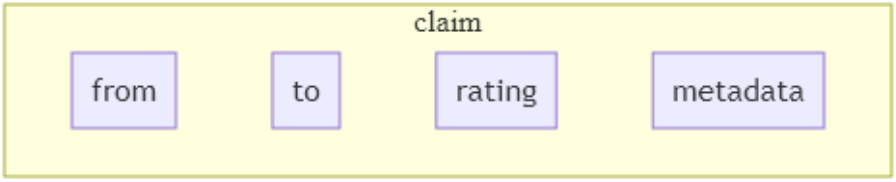


Figure 2: The Structure of Claim

$$claim = \langle identifier_{\texttt{from}}, identifier_{\texttt{to}}, rating, metadata \rangle$$

trustor an *entity* that may trust or distrust the other *entity* (the *trustee*) or evaluate the *rating* of the *identity document* of the other *entity* for

[10] https://tools.ietf.org/html/rfc7070

[11] Formalising Trust as a Computational Concept, Stephen Paul Marsh, Department of Computing Science and Mathematics, University of Stirling

[12] http://theworld.com/~cme/html/trust.html

making an decision. Trustor may be a social agent (such as a person or an institution) or a technical agent (such as a computer or a software application), acting on behalf of a social agent.[13] [14] [15] A *trustor* may be called a verifier.

trustee an *entity* that may prove its identity to the *trustor*.

recommender an *entity* that recommend the *trustee* to the *trustor* through *claim*s.

Figure 3: Trustor & Trustee & Recommender

trust graph a weighted directed graph about trust or authorization relations among the identities consists of a set of *identity document*s and a set of *claim*s. We use $\mathcal{G}$ to denote the *trust graph*. $\mathcal{V}$ is the vertices of the *trust graph*, which consists of a set of *identity document*s. $\mathcal{E} = \langle i, j \rangle$ is the edges of the *trust graph* where $i, j \in \mathcal{V}$, which consists of a set of pairs of *identity document*s that the *identifier*s in *claim*s point to. And $\mathbf{A}$ is the weight adjacency matrix of the *trust graph* where $a_{i,j}$ denotes the scalar entry in the (i, j)-th position of matrix, representing the *rating* in the *claim* from i to j. Specially, we specify $a_{i,j} = +1$ if $i = j$.

$$\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A} \rangle$$

IDnet an online subgraph of *trust graph* stored on NEO blockchain. In NEO ID, not all of the *identity document*s and *claim*s are on chain. Actually a lot of *identity document*s and *claim*s are offline and held by their owners. But *identity document*s and *claim*s can be easily traceable and revokeable if they are on chain. So the *IDnet* may be the backbone network of global NEO ID *trust graph*. We use $\mathcal{G}_{\mathbf{ID}}$ to denote the global *IDnet*. If on-chain and off-chain information are in conflict, the truth is defaulted to the on-chain values.

trust profile an offline subgraph of *trust graph* about a *trustor* that consists of the set of *recommender*s that the *trustor* trust and the *claim*s from the *trustor* to each *recommender*s. We use $\mathcal{G}_{+i}$ to denote the *trust profile* of i.

trust proof an offline subgraph of *trust graph* about a *trustee* that consists of the set of *recommender*s that trusts the *trustee* and the *claim*s from each *recommender*s to the *trustee*. We use $\mathcal{G}_{-i}$ to denote the *trust proof* of i.

[13] https://en.wikipedia.org/wiki/Trustor_(agent)
[14] Hardin R. (2002) Russel Hardin: Trust and trustworthiness. Russel Sage Foundation.
[15] Cofta, P. (2007) Trust, Complexity and Control: Confidence in a Convergent World. John Wiley and Sons.

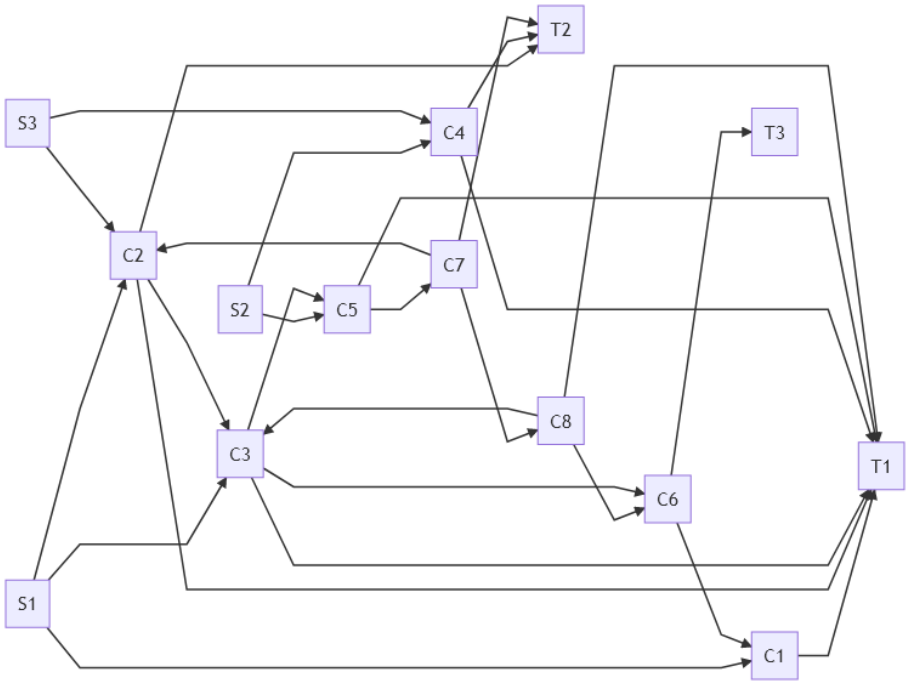


Figure 4: Example of Trust Graph

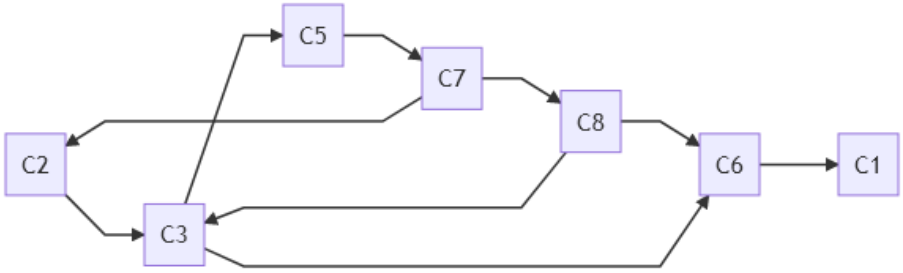


Figure 5: Example of IDnet

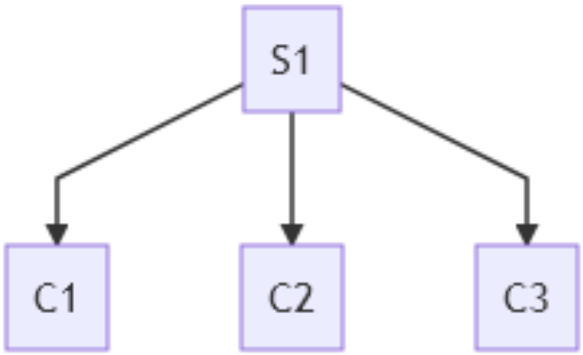


Figure 6: Example of Trust Profile of S1

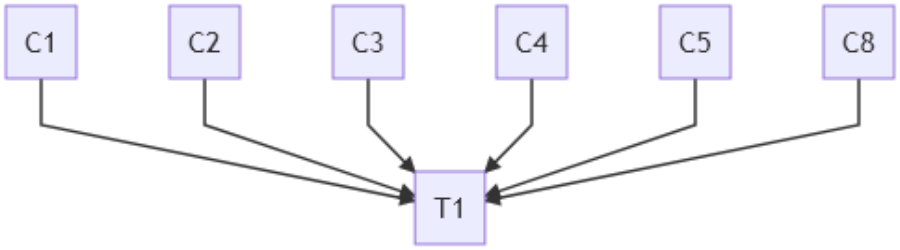


Figure 7: Example of Trust Proof of T1

rating evaluation function a function $f_{\mathcal{G}} : (x, y) \rightarrow [-1, +1]$ to solve an overall rating metric from the *trustor* x to the *trustee* y on the *trust graph* $\mathcal{G}$. We will continue discussing about the *rating evaluation function* in the analysis section.

NEO ID Operations

This section describes the operations of NEO ID include online operations (smart contract) and offline operations.

There are two kinds of operations: operations on *identity document* and operations on *claim*. Good implementation may also supply more high level operations based on the following basic operations. For example,

- **delegate**: anyone can delegate others to operate his own identity document (include create, remove, update). Either because of the other party needs to bear the cost of the operation, or lack of token in account, or networking incapability (such as not having a online access), or some other situation needs(e.g., I should delegate my school to create an identity document which contains the proposal of my transcript and upload it to the blockchain

and then issue a claim to this identity document to support the proposal, Instead of creating a transcript by myself. Because I don't have the right to make a transcript).

- **permissions grading control**: anyone can delegate others to operate his own identity document (include create, remove, update). Either because of the other party needs to bear the cost of the operation, or lack of token in account, or networking incapability (such as not having a online access), or some other situation needs(e.g., I should delegate my school to create an identity document which contains the proposal of my transcript and upload it to the blockchain and then issue a claim to this identity document to support the proposal, Instead of creating a transcript by myself. Because I don't have the right to make a transcript).

Online Operations

An online operation T would be implemented by a smart contract on NEO blockchain. There may be some (gas) cost to perform such operations, but the online operations will update the global state machine (the *IDnet*) of NEO ID, providing more features such as traceability and revocability.

IDnet $\mathcal{G}_{\mathbf{ID}}^{(t)}$ will changed to $\mathcal{G}_{\mathbf{ID}}^{(t+1)}$ after an online operation T.

$$\mathcal{G}_{\mathbf{ID}}^{(t+1)} = T(\mathcal{G}_{\mathbf{ID}}^{(t)})$$

Identity Document

An online operation of *identity document* $T_{\mathbf{v}}$ changes an *identity document* of the *IDnet* $\mathcal{G}_{\mathbf{ID}} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A} \rangle$.

create Anyone can create an *identity document* of itself and upload it to the blockchain. The identifier of the new *identity document* must not already exist. A new vertex v is added to the *IDnet* with no edge added.

The *identity document* may be added to the *IDnet* only by the owner itself, but some high level operations in some implementations may also supply some operations to delegate another one to add the *identity document* on *IDnet*.

$$v \notin \mathcal{V} \to T_{\mathbf{v}}^{\mathbf{add}(v)}(\mathcal{G}_{\mathbf{ID}}) = \langle \mathcal{V} \cup \{v\}, \mathcal{E}, \begin{bmatrix} \mathbf{A} & \\ & a_{v,v} \end{bmatrix} \rangle$$

remove Anyone can remove an existed online *identity document* of itself. The identifier of the *identity document* is needed. But it is expensive and not recommended due to all *claim*s to and from this *identity document* needing to be destroyed by the claim-remove operation. This operation may cause gas exhausted in NEO smart contract because one may issue or be issued a lot of *claim*s and the issued *cliam*s are out of its control (one

can issue a claim to any other one at any time). Actually this operation cannot wipe your data on blockchain because the data structure of NEO block is append-only, in other word, this operation only mark your *identity document* leave the *IDnet*

The *identity document* may be removed from the *IDnet* only by the owner itself, but some high level operations in some implementations may also supply some operations to delegate another one to remove the *identity document* on *IDnet*.

$$v \in \mathcal{V} \to T_{\mathbf{v}}^{\mathbf{del}(v)}(\mathcal{G}_{\mathbf{ID}}) = \langle \mathcal{V} \setminus \{v\}, \{\langle i,j\rangle | \langle i,j\rangle \in \mathcal{E}, i \neq v, j \neq v\}, \mathbf{J}_{v,\mathcal{V}} \cdot \mathbf{A} \cdot \mathbf{J}_{v,\mathcal{V}}^{T} \rangle$$

update Anyone can update an online *identity document* of itself using the document's identifier. But it is expensive and not recommended since it involves first a remove then a create operation with the remove operation involving multiple claim removal operations. It is recommended to create a new *identity document* instead. The data is also not actually destroyed since all data is append-only on the blockchain.

The *identity document* may be updated in the *IDnet* only by the owner itself, but some high level operations in some implementations may also supply some operations to delegate another one to update the *identity document* on *IDnet*.

$$x \in \mathcal{V}, y \notin \mathcal{V} \to T_{\mathbf{v}}^{\mathbf{set}(x,y)}(\mathcal{G}_{\mathbf{ID}}) = T_{\mathbf{v}}^{\mathbf{add}(y)}(T_{\mathbf{v}}^{\mathbf{del}(x)}(\mathcal{G}_{\mathbf{ID}}))$$

Claim

An online operation of *claim* $T_{\mathbf{e}}$ changes an *claim* of the *IDnet* $\mathcal{G}_{\mathbf{ID}} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A} \rangle$.

create Anyone with an online *identity document* can issue a claim to another one to support the proposal. The identifier of the *identity document* containing the supported proposal and the rating of the claim is needed. The proposals of the supporter x and supportee y should both be verified according to the implementation's scheme. Possible proposal verification methods are discussed in the analysis section. This operation adds an edge $e = \langle x, y \rangle$ from *identity document* x to *identity document* y with rating a to the *IDnet*.

The *claim* may be added to the *IDnet* only by the issuer itself, but some high level operations in some implementations may also supply some operations to delegate another one to add the *claim* on *IDnet*.

$$x, y \in \mathcal{V}, \langle x, y \rangle \notin \mathcal{E} \to T_{\mathbf{e}}^{\mathbf{add}(e,a)}(\mathcal{G}_{\mathbf{ID}}) = \langle \mathcal{V}, \mathcal{E} \cup \{e\}, \mathbf{A} + \begin{bmatrix} & & \\ & a - a_{x,y} & \\ & & \end{bmatrix} \rangle$$

remove Anyone can remove a claim it issued before to another *identity document*. The identifier of the previously-supported *identity document* is needed. This operation removes an edge $e = \langle x, y \rangle$ from the *IDnet*.

The *claim* may be removed from the *IDnet* only by the issuer itself, but some high level operations in some implementations may also supply some operations to delegate another one to remove the *claim* on *IDnet*.

$$x, y \in \mathcal{V}, \langle x, y \rangle \in \mathcal{E} \to T_{\mathbf{e}}^{\mathbf{del}(e)}(\mathcal{G}_{\mathbf{ID}}) = \langle \mathcal{V}, \mathcal{E} \setminus \{e\}, \mathbf{A} + \begin{bmatrix} & & \\ & -a_{x,y} & \\ & & \end{bmatrix} \rangle$$

update Anyone can update a claim it issued before. The identifier of the supported *identity document* and the updated rating is needed. This operation updates the *rating* to be a from *identity document* x to *identity document* y. Since the proposals of x and y should have already been verified during the claim creation operation, optimized implementations may not need to verify the respective proposals during the update operation.

The *claim* may be updated in the *IDnet* only by the issuer itself, but some high level operations in some implementations may also supply some operations to delegate another one to update the *claim* on *IDnet*.

$$x, y \in \mathcal{V}, \langle x, y \rangle \ \mathcal{E} \to T_{\mathbf{e}}^{\mathbf{set}(e,a)}(\mathcal{G}_{\mathbf{ID}}) = \langle \mathcal{V}, \mathcal{E}, \mathbf{A} + \begin{bmatrix} & & \\ & a - a_{x,y} & \\ & & \end{bmatrix} \rangle$$

Offline Operation

An offline operation would be implemented by a sdk or something else. Performing such operations may have little cost. But it increases the complexity for tracing, revoking and many other features.

It is a good suggestion that you may add a valid time or expire time in the *proposal* of offline *identity document* to reduce the risk because it is diffcult to revoke.

create offline signed identity document anyone can generate its offline *identity document* by itself with the folowing steps. 1. generate a asymmetric cryptographic key pair 2. generate a valid *identifier* point to the *identity document* 3. write a *proposal* of the entity itself 4. write some addtional infomation in the *metadata* 5. combine the *identifier*, *public key*, *proposal* and *metadata* mentioned above and calculate a signature with the private key and hold the private key.

create offline signed claim if some one has an *identity document*, it can issue an offline signed claim with the following steps. 1. use the *identifier*

itself as the from *identifier* 2. check the *identity document* of another one especially the *proposal* and evaluate a *rating* and use the *identifier* as the to *identifier* 3. write some addtional infomation in the *metadata* 2. combine the from *identifier*, the to *identifier*, the *rating* and the *metadata* mentioned above and calculate a signature with its private key.

create offline certificate we can combine the *create offline signed identity document* and *create offline signed claim* operations, to produce a offline certificate.

NEO ID Retrieving

Anyone can retrieve data from the NEO ID platform. There are two basic retrieval methods stipulated in NEO ID but implementations may have more functions such as searching / filtering methods to improve the performance and ease of use. These functions may be able retrieve *identity documents* or claims that fit a criteria. For example, functions can be implemented that return:

1. all claims that have been issued by an *identity document*
2. all claims that have been issue to an *identity document*
3. the shortest path betweeen two *identity documents*
4. all *identity documents* that have issued claims with positive ratings to a specified *identity document*.

Get Identity Document by Identifier

Retrieval of an *identity document* by its identifier (DID). This basic operation is used by any entity wishing to view the *identity document* of another entity for uses such as:

1. a trustee reviewing its own *identity document* data
2. a trustor verifying another entity's proposals or reading the metadata.
3. a CA evaluating an entity's proposals to determine a rating for issuing a claim to its *identity document*

Get Claim by Identifier Pair

Retrieval of a claim by its origin and destination *identity documents*. This basic operation is used by any entity wishing to view the trust relationship between two entities, including between itself and other entities. The retrieved claims can be used for:

1. a trustor to evaluate or re-evaluate the trustworthiness of an *identity document*
2. a CA to update the rating to an *identity document*
3. an *identity document* to view its rating from another *identity document*

NEO ID Procedure

We now present three examples about the issuing, revoking/updating and verifying procedures of NEO ID. Implementations may have their own specific methods for these procedures. But they should be in accordance with the protocols and design outlined in this whitepaper.

Example of Issuing Online NEO ID

The following are the steps for getting an online NEO ID.

1. (Optional) The entity request a CA for an ID.
2. (Optional) The entity negotiates the contents of the identity until both the entity and CA are in agreement. This ensures the CA will verify the entity's proposals and issue a claim.
3. The entity creates an *identity document* which includes a *proposal* about it (may be an *identity document* negotiated in step 2) on *IDnet* via `identity document - create` operation.
4. The entitiy requests a claim from the CA.
5. The CA retrieves the relevant *identity document* from *IDnet*.
6. The CA evaluates the *proposal* and determines a *rating*.
7. If the CA accepts the *identity document*, it submits a claim for that *identity document* to *IDnet* via `claim - create` operation.
8. The CA notifies the entity of success if it accepts the *identity document*, or notifies the entity of failure else.
9. The entity repeats steps 4-8 to request *claim*s from other CAs.

Example of Issuing Offline NEO ID

The following are the steps for getting an offline NEO ID.

1. (Optional) The entity request a CA for an ID.
2. (Optional) The entity negotiates the contents of the identity until both the entity and CA are in agreement. This ensures the CA will verify the entity's proposals and issue a claim.
3. The entity generate an offline *identity document* which includes a *proposal* about it (may be an *identity document* negotiated in step 2).
4. The entitiy requests a claim from the CA.
5. The CA evaluates the *proposal* and determines a *rating*.
6. If the CA accepts the *identity document*, it generate a claim for it and sign the claim.
7. The CA send the signed claim to the entity if it accepts the *identity document*, or notifies the entity of failure else.

Example of Revoking / Updating

The following are the steps for revoking/updating a NEO ID:

1. (Optional) The entity request the CA for revokeing/updating a claim.

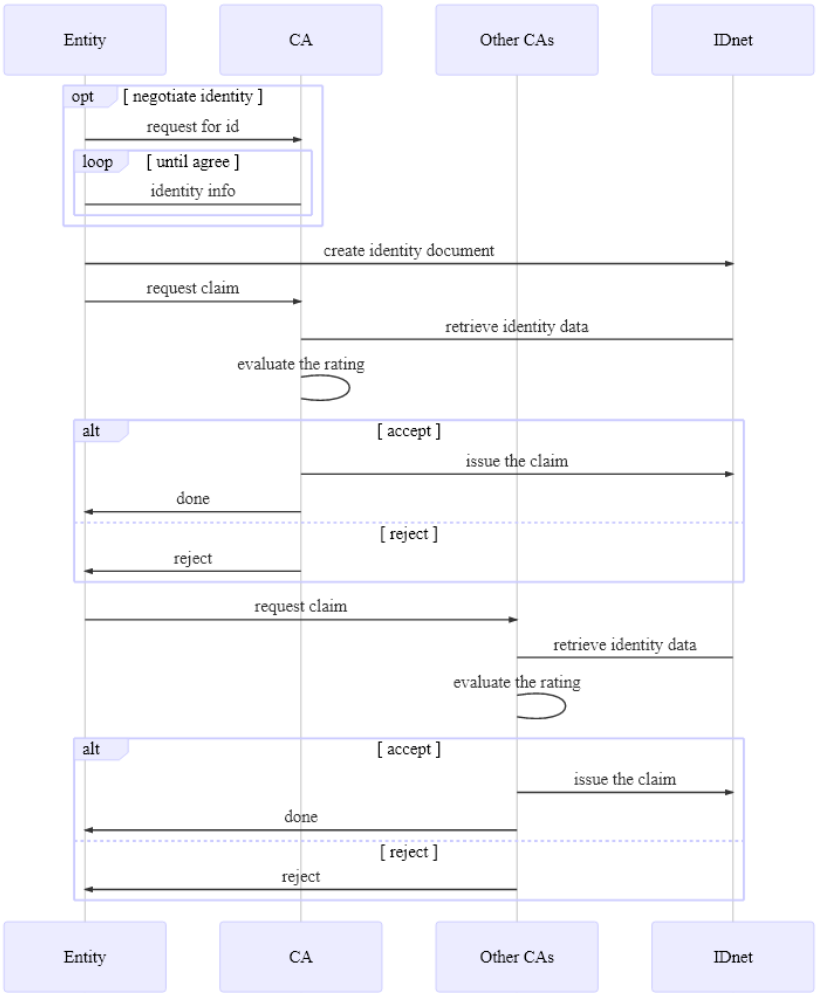


Figure 8: Example of getting an online NEO ID

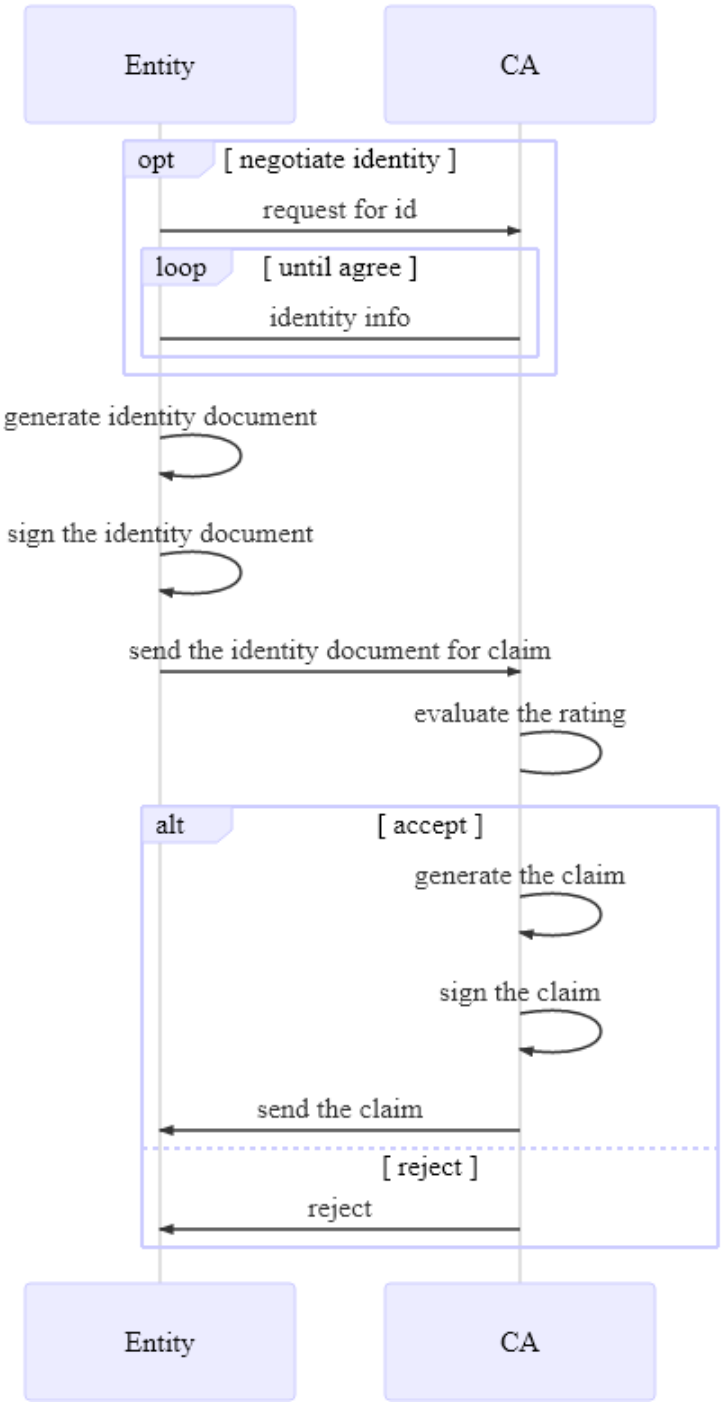


Figure 9: Example of getting an offline NEO ID

2. The CA revokes/updates the claim on the IDnet either of its own accord or prompted by an entity's request.
3. (Optional) The CA notifies the entity of the claim revoking/updating.

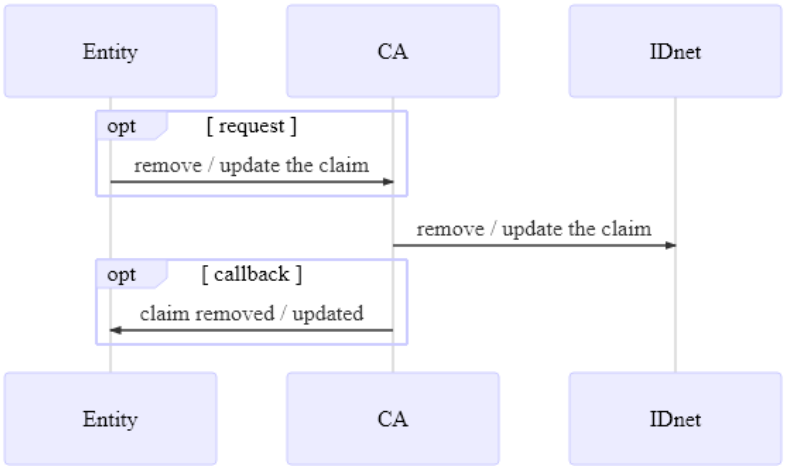


Figure 10: Example of revoking/updating a NEO ID

Specially, if a CA want to revoke an offline *claim*, it can set an *claim* with *rating* 0 on the IDnet which override the offline *claim*.

Example of Verifying

The following are the steps for verifying a NEO ID:

1. The trustor/verifier requests an entity for a trust proof.
2. The entity submits a trust proof to the trustor/verifier.
3. The trustor/verifier validates the trust proof's format, signatures and *proposal*s, etc.
4. The trustor/verifier loads its trust profile.
5. The trustor/verifier retrieves the necessary information from *IDnet* to build a local *trust graph*.
6. The trustor/verifier simplifies the local *trust graph*.
7. The trustor/verifier evaluates the *rating* of the entity based on the local *trust graph* via its *rating evaluation function*.

Privacy

The security and privacy of an entity's identity information is of utmost importance and a core tenant of any decentralized identity system. According to the W3C DID working group's report, strong privacy features are especially critical for a global identity system that uses immutable public blockchains. DID architecture can incorporate Privacy by Design at the very lowest levels of

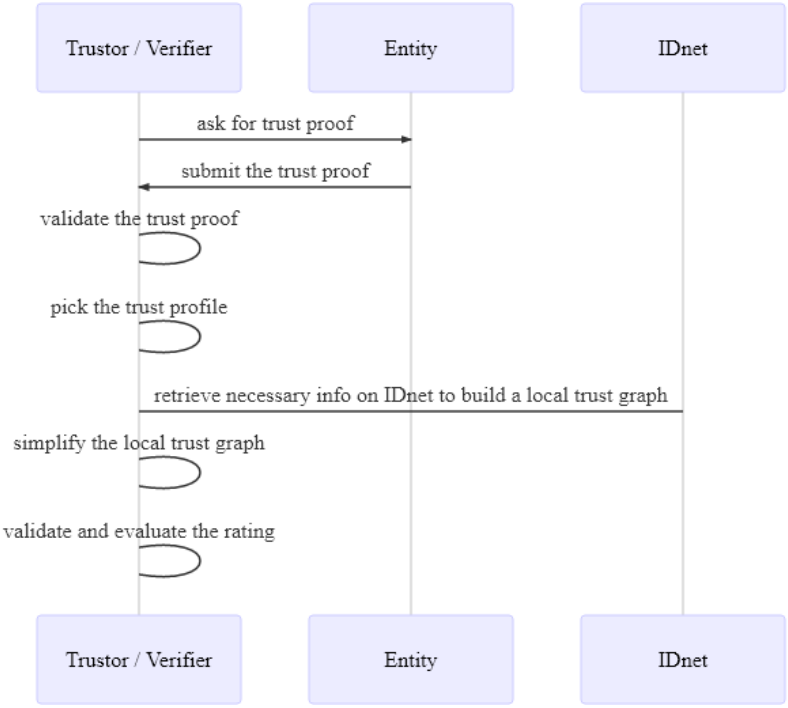


Figure 11: Example of verifying a NEO ID

infrastructure and thus become a powerful, new, privacy-preserving technology if deployed using best practices such as:

1. Pairwise-pseudonymous DIDs. While DIDs can be used as well-known public identifiers, they can also be used as private identifiers issued on a per-relationship basis. So rather than a person having a single DID, like a cell phone number or national ID number, he can have thousands of pairwise-unique DIDs that cannot be correlated without his consent, yet can still be managed as easily as an address book. The DID Document for a pseudonymous DID SHOULD also use pairwise-unique public keys. It might seem natural to also use pairwise-unique service endpoints in the DID Document for a pseudonymous DID. However, unique endpoints allow all traffic between to DIDs to be isolated perfectly into unique buckets, where timing correlation and similar analysis is easy. Therefore, a better strategy for endpoint privacy may be to share an endpoint among thousands or millions of DIDs controlled by many different subjects.

2. Off-chain private data. Storing any type of PII on a public blockchain, even encrypted or hashed, is dangerous for two reasons: 1) the encrypted or hashed data is a global correlation point when the data is shared with multiple parties, and 2) if the encryption is eventually broken (e.g., quantum computing), the data will be forever accessible on an immutable public ledger. So the best practice is to store all private data off-chain and exchange it only over encrypted, private, peer-to-peer connections. All PII should be kept behind service endpoints under the control of the subject. With this privacy architecture, PII may be exchanged on a private, peer-to-peer basis using communications channels identified and secured by key descriptions in DID documents. This also enables subjects and relying parties to implement the GDPR right to be forgotten, as no PII will be written to an immutable ledger.

3. Selective disclosure. The decentralized PKI (DPKI) that DIDs make possible opens the door to individuals gaining greater control over their personal data in two ways. First, it enables it to be shared using encrypted digital credentials. Second, these credentials can use zero-knowledge proof cryptography for data minimization, e.g., you can disclose that you are over a certain age without disclosing your exact birthdate.

4. Herd Privacy. When a DID subject is indistinguishable from others in the herd, privacy is available. When the act of engaging privately with another party is by itself a recognizable flag, privacy is greatly diminished. DIDs and DID methods SHOULD work to improve herd privacy, particularly for those who legitimately need it most. Choose technologies and human interfaces that default to preserving anonymity and pseudonymity. In order to reduce digital fingerprints, share common settings across client implementations, keep negotiated options to a minimum on wire protocols, use encrypted transport layers, and pad messages to standard lengths.

Privacy of Identity Documents

To achieve the aforementioned privacy practices for *identity documents*, we can describe one implementation that uses various privacy-protecting technologies:

Each entity may simultaneously possess multiple *identity documents* to describe its various identities and also to serve its various roles on the IDnet. One entity may have an identity document listing its attributes related to education, such as its degrees completed or skill certificates. It would show this *identity document* to relevant verifiers such as employers. However, this entity could also have another *identity document* that describes its attributes linked to open-source project contributions that it could present to the moderator of a developer community. An entity would be able to only show a verifier an identity document relevant to the its needs and this data compartmentalization helps ensure the selective disclosure of the entity's data. Since we separate the proposals of an *identity document* into scopes and attributes, we define scopes to be public and attributes to be private. To ensure the privacy of an entity's data, all *identity documents* with attributes are not kept on the blockchain since the loss of a private key or future developments in quantum computing could render the data accessible to unwanted parties, even if the data is encrypted through hashing. The *identity documents* that serve as CAs or recommenders, however, are kept on the blockchain because these *identity documents* only contain public scopes and need to be discoverable by trustors building their trust profile.

All off-chain communication between trustors and CAs, trustors and trustees, and CAs and trustees can be conducted through secure peer-to-peer channels with end-to-end asymmetric or symmetric encryption.

Privacy of Claims

To achieve the aforementioned privacy practices for claims, we can describe one implementation that uses various privacy-protecting technologies:

Each *identity document* may simultaneously possess multiple claims from different CAs to support its proposals. An *identity document* could have a claim fro a city-level government office and a national-level government office. It would only need to show its state-level claim to a verfifier needing that level of confirmation. An *identity document* would be able to only show a verifier a claim relevant to the its needs and this data compartmentalization helps ensure the selective disclosure of the entity's data.

When a new *identity document* is added to the IDnet, it might submit its proposals to various CAs for verification and support. This verification can be conducted through the use of zero-knowledge proofs or homomorphic encryption to protect the privacy of the proposals. If the CA recognizes the *identity document's* proposals, it then provides the *identity document* with a claim. To ensure that the *identity document* does not alter its proposals after obtaining a claim, the hash of the *identity document* in its approved state is appended

to the claim to guarantee immutability. All off-chain communication between trustors and CAs, trustors and trustees, and CAs and trustees can be conducted through secure peer-to-peer channels with end-to-end asymmetric or symmetric encryption.

Specification

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.[16]

Implementation format is not mandatory, the following specifications are only recommended practices.

Identity document

An *identity document* consists of an *identifier*, at least one *public key*, a *proposal* and a *metadata*, it MAY include other properties, such as signature, service and authentication. And an *identity document* is corresponding to a single *identifier*.

An *identity document* MUST be a single JSON object conforming to RFC8259.[17] You can use various data representation syntaxs to implement it, such as JXD, XML, JSON, JSON-LD. It is RECOMMENDED to use JSON[18] and JSON-LD.[19]

An *identity document* SHOULD be like this:

```
{
    "id": ...,
    "publicKey": [ ... , ... ],
    "proposal":  ... ,
    "metadata":  ... ,
    ...
}
```

identifier

Identifier MUST be globally unique. It is RECOMMENDED that your naming scheme SHOULD follow the rules below.[20]

```
identifier          = "did:" method-name ":" method-specific-id
method-name         = 1*method-char
method-char         = %x61-7A / DIGIT
```

[16] https://tools.ietf.org/html/rfc2119
[17] https://tools.ietf.org/html/rfc8259
[18] https://tools.ietf.org/html/rfc8259
[19] https://www.w3.org/TR/json-ld/
[20] https://w3c-ccg.github.io/did-spec/#generic-did-syntax

```
method-specific-id = *idchar *( ":" *idchar )
idchar             = ALPHA / DIGIT / "." / "-" / "_"
```

For example, a naming scheme MAY be like this:

```
identifier         = "did:neo:" network ":" specific-idstring
network            = ("test"/"main")
specific-idstring  = NEO public address
```

And, a specific identifier MAY be like this:

```
did:neo:03ada688020f7603e84c3fef6de2eaa5c658ca35b0e61dfa09330f06bd8a3b023f
```

public key

Public key is used for digital signatures, encryption and other cryptographic operations, the value of the publicKey property MUST be an array of public keys. The rules for public keys are:[21]

1. The value of the publicKey property MUST be an array of public keys.
2. Each public key MUST include id and type properties, and exactly one value property.
3. The array of public keys SHOULD NOT contain duplicate entries with the same id and different value properties with different formats.
4. Each public key MUST include a controller property, which identifies the controller of the corresponding private key.
5. The value property of a public key MUST be exactly one of publicKeyPem, publicKeyJwk, publicKeyHex, publicKeyBase64, publicKeyBase58, publicKeyMultibase, depending on the format and encoding of the public key.

For example:

```
{
    "id": "did:neo:03ada688020f7603e84c3fef6de2eaa5c658ca35b0e61dfa09330f06bd8a3b023f#keys-1
    "type": "Ed25519VerificationKey2018",
    "controller": "did:neo:03ada688020f7603e84c3fef6de2eaa5c658ca35b0e61dfa09330f06bd8a3b023
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJdfywnZn6z3wXmqPV"
}
```

proposal

Proposal is RECOMMENDED to implement by metaprogramming language drived by symbolic calculation such as `ML`[22], `LISP`, `YACC`[23] or `Backus-Naur`

[21] https://w3c-ccg.github.io/did-spec/#public-keys

[22] The Definition of Standard ML, Robin Milner, Mads Tofte, Robert Harper, MIT Press 1990; (revised edition adds author David MacQueen), MIT Press 1997, ISBN 0-262-63181-4.

[23] "The A-Z of Programming Languages: YACC". Computerworld. Retrieved 30 November 2012.

`form`[24] because it is an expression to describe the set of attribute of the entity or the authorization limit scope of the entity. The proposal should be automatic verified or proved in limited computing resource and limited time.

metadata

There are no specific requirements for metadata, but it MUST meet the format requirements of the JSON object.

For example:

```
{
    "create time": "18:00"
}
```

Claim

A *claim* consists of two *identifiers* of *identity documents*, a *rating* and a *metadata*. It MAY include other properties, such as signature. It should be noted that a *claim* without a signature SHALL NOT be an effective *claim*.

A *claim* MUST be a single JSON object conforming to RFC8259. You can use various data representation syntaxs, such as JXD, XML, JSON, JSON-LD. It is RECOMMENDED to use JSON and JSON-LD.

An *claim* SHOULD be like this:

```
{
    "from":  ... ,
    "to":  ... ,
    "rating":  ... ,
    "metadata":  ... ,
    ...
}
```

The format of "from" identifier, "to" idebtifier and metadata SHOULD be the same as mentioned above in the *identity document* paragraph. Specially if the *identity document* is offline, the *claim* SHOULD contain the hash of the *identity document*.

rating

The *rating* attribute is expressed as a floating-point number between -1 and 1 inclusive. And 1 implies full agreement with the *proposal* in the *identity document*, while -1 indicates opposition to that.

For example:

[24] Morrison, Kelly (July 2, 1993), "Backus Normal Form vs. Backus-Naur Form", comp.compilers (newsgroup), IECC posting quotes Alan J. Perlis and Peter Naur from "ALGOL Section" of Richard L. Wexelblat, editor, History of Programming languages (1981)

`0.6`

Also, an *identity document* and a *claim* can be combine as an X.509 certificate following RFC5280.[25]

Analysis

Here we give some analysis of NEO ID including how the NEO ID works and some examples for implementation.

Trust Model

Firstly, the most important model of NEO ID is the trust model.

Trust is a common and essential phenomenon. Indeed, it has been argued that we as humans would not even be able to face the complexities of the world without resorting to trust, because it is with trust that we are able to reason sensibly about the possibilities of everyday life.[26]

Let us first define trust with a commonly accepted definition:

Trust (or conversely, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of our capacity of ever being able to monitor it) and in a context in which it affects [our] own action.[27]

From this definition, three properties of trust emerge: subjectiveness, context-dependence, and dynamism. The same behavior may lead to different trust levels in different trusting entities, hence subjectiveness qualifies trust. As trust (e.g., in giving good advice) in one context (e.g., academia) does not necessarily transfer to another context (e.g., industry), we add context-dependence to the list of trust properties. Finally, the fact that trust increases after successful observations, while it decays over time exemplifies its dynamism. As a result, trust evolution must embed the notion of time.[28]

Centralised protocols are protocols which use a well-known common trusted intermediary, call it the trusted authority(TA), to form a trust relationship between two mutually distrusting entities. For example, a Certificate Authority(CA) , an authority responsible for issuing and managing digital certificates, is as a trusted third party in e-commerce transactions, which is responsible for the legality of the public key in the public key system.

[25] https://tools.ietf.org/html/rfc5280

[26] Stephen Paul Marsh. Formalising Trust as a Computational Concept

[27] Gambetta, D. : Can We Trust Trust? In Trust: Making and Breaking Cooperative Relations, Gambetta, D (ed.)., Basil Blackwell, Oxford (1990).

[28] Daniele Quercia, Stephen Hailes, Licia Capra. B-trust: Bayesian Trust Framework for Pervasive Computing

The need to refer to a centrally trusted point poses a major problem as a general trust framework, because the assumed objectivity of trust will not work. In our definition of trust, trust is subjective. Furthermore, a TA can never be a good enough recommender for everyone in a large distributed system. Its credibility depletes, and its recommendations increase in uncertainty, as its community of trustees grows. A decentralised approach to trust management makes sense, which will complement centralized approaches to give a more complete general trust management framework.

For example, in real life, we use CA certification as a trust anchor and absolutely trust the CA. Trust does not rely on only one CA, but can disperse among multiple CAs. Once a CA has problems, such as the domain name being stolen, then the user cannot absolutely trust that CA.

With decentralization , each rational entity will be allowed to take responsibility for its own fate. Each entity then makes decisions for itself, on its own policies. The disadvantage of decentralisation is that more responsibility and expertise is required on the user for managing trust policies. However, entities on the network will still have the option of relying on centralised trust models so that this responsibility can be assigned to their trusted authority, if they wish to do so. Decentralisation does not completely replace current centralised approaches, but it gives entities a choice of managing their own trust.[29]

The Trust Model is important to NEO ID because it provides the tools needed to build a decentralized Public Key Infrastructure (PKI) for a trustor to evaluate the trust to a trustee. The Trust Model describes the relationship between entities, identity documents, claims, ratings and trust profiles. In the NEO ID system, each entity may simultaneously possess multiple identity documents that correspond to different identities (e.g. academic, employment, personal) and to different roles in the NEO ID system (e.g. trustor, recommender/CA, trustee). These identity documents can issue claims to each other and build local trust graphs used to evaluate the trust between one identity document and another. Identity documents contain proposals which detail the scope and attributes of an identity document. The design of proposals will be elaborated further in the proposal model section.

Rating

There are many ways to quantify trust, but to simplify our proposed model, we choose to represent trust as a continuous variable over a specific range $[-1, +1]$. So we denote *rating* $a \in [-1, +1]$ is a scalar to describe how much one entity trusts another. Here are some properties of *rating* a:

1. The maximum value of a is $+1$, which means blind trust.
2. The minium value of a is -1, which means complete distrust.
3. The trust value 0 is a neutral value, which means neither trust nor distrust.

[29] Abdul-rahman, Alfarez & Hailes, Stephen. (1997). Using Recommendations for Managing Trust in Distributed Systems.

4. The greater the value of a, the more trust exists from one entity to another.

Here is a possible stratification of trust values:

rating	label
$+1$	Blind Trust
$[+0.9, +1)$	Very High Trust
$[+0.75, +0.9)$	High Trust
$[+0.5, +0.75)$	High Medium Trust
$[+0.25, +0.5)$	Low Medium Trust
$(0, +0.25)$	Low Trust
0	No Trust or Distrust
$(-0.25, 0)$	Low Disrust
$(-0.5, -0.25]$	Low Medium Disrust
$(-0.75, -0.5]$	High Medium Disrust
$(-0.9, -0.75]$	High Disrust
$(-1, -0.9]$	Very High Disrust
-1	Complete Disrust

Trust Graph

A *trust graph* can be represented as a weighted directed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A} \rangle$. $\mathcal{V}$ is the vertices of the *trust graph*, which consists of a set of *identity document*s. $\mathcal{E} = \langle i, j \rangle$ is the edges of the *trust graph*, which consists of a set of pairs of *identity document*s that the *identifier*s in *claim*s point to. And $\mathbf{A}$ is the weight adjacency matrix of the *trust graph* where $a_{i,j}$ denotes the scalar entry in the (i, j)-th position of matrix, representing the *rating* in the *claim* from i to j. Specially, we specify $a_{i,j} = +1$ if $i = j$.

From the definition above, we know:

1. All the trust values in the weight adjacency matrix A are in the domain of $[-1, +1]$.
$$i, j \in \mathcal{V} \rightarrow a_{i,j} \in [-1, +1]$$
2. If i remains neutral about j, the trust value is 0.
$$\langle i, j \rangle \notin \mathcal{E} \rightarrow a_{i,j} = 0$$

Here we define some calculation operations on *trust graph* similar to the NEO ID operations:

1. add an *identity document* to the *trust graph*

The following equation describes adding a vertex v to the trust graph $\mathcal{G}$. The set of vertices $\mathcal{V}$ adds vertex v. The set of edges $\mathcal{E}$ remains unchanged. The matrix of weights A adds only $a_{v,v}$ with rating 1 meaning the v has full trust to itself.

$$\mathcal{G} + v = \langle \mathcal{V} \cup \{v\}, \mathcal{E}, \begin{bmatrix} \mathbf{A} & \\ & a_{v,v} \end{bmatrix} \rangle$$

2. remove an *identity document* from the *trust graph*

The following equation describes removing a vertex v from the trust graph $\mathcal{G}$. The set of vertices $\mathcal{V}$ removes vertex v. The set of edges $\mathcal{E}$ removes all edges to and from v. The matrix of weights A removes the row and column containing weights to and from v.

$$\mathcal{G} - v = \langle \mathcal{V} \setminus \{v\}, \{\langle i,j\rangle | \langle i,j\rangle \in \mathcal{E}, i \neq v, j \neq v\}, \mathbf{J}_{v,\mathcal{V}} \cdot \mathbf{A} \cdot \mathbf{J}_{v,\mathcal{V}}^T \rangle$$

3. add a *claim* from x to y with *rating* a

The following equation describes adding an edge from x to y with weight a to the trust graph $\mathcal{G}$. The set of vertices $\mathcal{V}$ remains unchanged. The set of edges $\mathcal{E}$ adds the edge from x to y. The matrix of weights A removes the previous weight from x to y and adds the new weight a.

$$\mathcal{G} + \langle \langle x,y\rangle, a\rangle = \langle \mathcal{V}, \mathcal{E} \cup \{\langle x,y\rangle\}, \mathbf{A} + \begin{bmatrix} & & \\ & a - a_{x,y} & \\ & & \end{bmatrix} \rangle$$

4. remove a *claim* from x to y

The following equation describes removing an edge from x to y from the trust graph $\mathcal{G}$. The set of vertices $\mathcal{V}$ remains unchanged. The set of edges $\mathcal{E}$ removes the edge from x to y. The matrix of weights A removes the weight $a_{x,y}$.

$$\mathcal{G} - \langle \langle x,y\rangle, a\rangle = \langle \mathcal{V}, \mathcal{E} \setminus \{\langle x,y\rangle\}, \mathbf{A} + \begin{bmatrix} & & \\ & -a_{x,y} & \\ & & \end{bmatrix} \rangle$$

5. merge a new *trust graph* $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A}\rangle$ with an old *trust graph* $\mathcal{G}_0 = \langle \mathcal{V}_0, \mathcal{E}_0, \mathbf{A}_0\rangle$

The following equation describes merging a new trust graph $\mathcal{G}$ with old trust graph $\mathcal{G}_0$. The set of vertices $\mathcal{V}$ and $\mathcal{V}_0$ will be unioned. The set of edges $\mathcal{E}$ and $\mathcal{E}_0$ will be unioned with any conflicts resolved by deferring to the new edges in $\mathcal{E}$ as accurate. The matrix of weights A merge the matrix of weights A_0 with conflicts resolved by deferring to the new weights in A as accurate

$$\mathcal{G} + \mathcal{G}_0 = \langle \mathcal{V} \cup \mathcal{V}_0, \mathcal{E} \cup \mathcal{E}_0, \mathbf{A} + \mathbf{A}_0 \rangle$$

Rating Evaluation Function

rating evaluation function a function $f_{\mathcal{G}} : (x,y) \to [-1,+1]$ to solve an overall rating metric from the *trustor* x to the *trustee* y on the *trust graph* $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathbf{A}\rangle$. Each application / trustor / implementation of the NEO ID is able to determine their own *rating evaluation function* to satisfy their needs.

But here we have the following assumptions about the trust evaluation function f:

1. Trust will gradually decay during the process of transition

The rating evaluation function's value from x to v will never increase if a new edge from y to v is added to the trust graph $\mathcal{G}$ containg an edge from x to y.

$$\frac{f_{\mathcal{G}+\mathcal{G}_0}(x,v)}{f_{\mathcal{G}}(x,y)} \in [-1,+1]$$

```
{.mermaid caption="Example of Trust Decay 1"}    graph LR     X[X]
G((part of trust graph))     Y[Y]     V[V]     X --> G     G -->
Y     Y -.-> V
```

The rating evaluation function's value from v to y will never increase if a new edge from v to x is added to the trust graph $\mathcal{G}$ containg an edge from x to y.

$$\frac{f_{\mathcal{G}+\mathcal{G}_0}(v,y)}{f_{\mathcal{G}}(x,y)} \in [-1,+1]$$

```
{.mermaid caption="Example of Trust Decay 2"}    graph LR     X[X]
G((part of trust graph))     Y[Y]     V[V]     V -.-> X     X -->
G     G --> Y 2. More positive trust relations never decrease the overall trust
in the network. More negative trust relations never increase the overall trust in
the network.
```

Adding an edge from x to y with a positive weight will never decrease the rating evaluation function's value from x to y. Adding an edge from x to y with a negative weight will never increase the rating evaluation function's value from x to y.

$$f_{\mathcal{G}_0}(x,y)(f_{\mathcal{G}+\mathcal{G}_0}(x,y) - f_{\mathcal{G}}(x,y)) \geqslant 0$$

```
{.mermaid caption="Example of Trust Combination"}    graph LR
X[X]     G((part of trust graph))     Y[Y]     V[V]     X --> G
G --> Y     X -.-> V     V -.-> Y
```

Discussion of Rating Evaluation Function

In the following section, we discuss the factors to consider in developing a rating evaluation function as well as the limitations of proposed solutions. These discussions are meant as a reference for developing trust functions and are not necessarily optimal solutions.

We begin with a simple case and define a few terms:

Normal trust values are only between $[0,+1]$ so there are no negative trust relations;

Extended trust values can span the range [-1,+1];

Direct the trust function only evaluates trust between adjacent entities;

Recommended the trust function evaluates trust between entities separated by multiple hops;

Reliable trust is absolute, either fully trusting or not, and without gradient levels in between;

Unreliable the trust value is a continuous variable between [-1,+1];

Passive the trustee (the entity whose identity is being verified) seeks out and assembles the information for determining its trust and is able to discard negative information;

Active the trustor (the entity who is verifying the identity of a target trustee) seeks out and assembles all relevant trust information regarding the trustee;

1. Normal Direct Reliable Trust In this basic case, we only trust adjacent entities connected by an edge of trust value +1 or remain neutral. We ignore all the trust value $a \in [-1, +1)$. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{NDRT}}(x, y) = \begin{cases} 1 & a_{x,y} = 1 \\ 0 & \text{else} \end{cases}$$

2. Extended Direct Reliable Trust Similar to Normal Direct Reliable Trust, with the addition of allowing overall trust value -1. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{EDRT}}(x, y) = \begin{cases} +1 & a_{x,y} = +1 \\ -1 & a_{x,y} = -1 \\ 0 & \text{else} \end{cases}$$

3. Normal Direct Reliable Threshold Trust Based on Normal Direct Reliable Trust, with the addition of a threshold $a_T \in (0, 1)$ to decide whether an entity should be trusted. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{NDRTT}}(x, y) = \begin{cases} 1 & a_{x,y} \geq a_T \\ 0 & \text{else} \end{cases}$$

4. Extended Direct Reliable Threshold Trust Based on Extended Direct Reliable Trust, with the addition of a threshold $a_T \in (0, 1)$ to decide whether an entity should be trusted or distrusted. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{EDRTT}}(x, y) = \begin{cases} +1 & a_{x,y} \geq a_T \\ -1 & a_{x,y} \leq -a_T \\ 0 & \text{else} \end{cases}$$

5. Extended Direct Unreliable Trust Based on Extended Direct Reliable Trust, except we can use the direct trust value as the overall trust metric. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{EDUT}}(x, y) = a_{x,y}$$

6. Normal Direct Unreliable Trust Based on Extended Direct Unreliable Trust, and now we ignore the trust value $a \in [-1, 0)$. Thus we have the following expression:

$$f_{\mathcal{G}}^{\mathbf{EDUT}}(x, y) = max(a_{x,y}, 0)$$

7. Normal Recommended Reliable Trust Based on Normal Direct Reliable Trust, with the additional assumption that trust can be passed from one entity to another so the trust function now evaluates trust between entities separated by hops.

The following equations means that in all possible permutations $\mathcal{P}$ of ordering the *identity document*s, if there exists an ordering $\mathbf{p}$ where x falls before y, then the trust value is evaluated or else it is 0. Here $\mathbf{p}_k$ is the kth *identity document* in the ordering $\mathbf{p}$.

$$f_{\mathcal{G},\mathbf{p}}^{\mathbf{NRRT}}(x, y) = \begin{cases} \prod_{k=i}^{j-1} f_{\mathcal{G}}^{\mathbf{NDRT}}(\mathbf{p}_k, \mathbf{p}_{k+1}) & i < j \\ 0 & \text{else} \end{cases}$$

`Given a permutation obeying the expression above exists, if the trust function evaluates to`

$$f_{\mathcal{G}}^{\mathbf{NRRT}}(x, y) = \begin{cases} 1 & \exists \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{NRRT}}(x, y) = 1 \\ 0 & \text{else} \end{cases}$$

8. Extended Passive Recommended Reliable Trust Similar to Normal Recommended Reliable Trust (since negative values are discarded by trustee) and Extended Direct Reliable Trust. We have the following expression:

$$f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x, y) = \begin{cases} \prod_{k=i}^{j-1} f_{\mathcal{G}}^{\mathbf{EDRT}}(\mathbf{p}_k, \mathbf{p}_{k+1}) & i < j \\ 0 & \text{else} \end{cases}$$

$$f_{\mathcal{G}}^{\mathbf{EPRRT}}(x, y) = \begin{cases} 1 & \exists \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x, y) = 1 \\ 0 & \text{else} \end{cases}$$

9. Extended Active Recommended Reliable Trust Based on Extended Passive Recommended Reliable Trust and Extended Direct Reliable Trust. The trust function returns 1 only when in all possible permutations of the ordering of identity documents from x to y, there exists an ordering where the trust function returns 1 and no orderings where the trust function

returns -1. Conversely, The trust function returns -1 only when in all possible permutations of the ordering of identity documents from x to y, there exists an ordering where the trust function returns -1 and no orderings where the trust function returns 1. In other cases, the trust function returns 0. We have the following expression:

$$f_{\mathcal{G}}^{\mathbf{EPRRT}}(x,y) = \begin{cases} +1 & \begin{cases} \exists \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x,y) = +1 \\ \forall \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x,y) \neq -1 \end{cases} \\ -1 & \begin{cases} \exists \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x,y) = -1 \\ \forall \mathbf{p} \in \mathcal{P}, f_{\mathcal{G},\mathbf{p}}^{\mathbf{ERRT}}(x,y) \neq +1 \end{cases} \\ 0 & \text{else} \end{cases}$$

10. Recommended Unreliable Trust

Now we propose an analogy to reach a solution for Recommended Unreliable Trust. First, we make a strong assumption that trust evaluation in a network can be calculated with two operations, passing and combining. We use $\times_*$ to denote the passing operation and Σ_* to denote the combining operation. According to the properties of *rating evaluation function* mentioned above: trust will gradually decay during the process of transition; more positive trust relations never decrease the overall trust in the network and more negative trust relations never increase the overall trust in the network, then we have

$$\frac{a \times_* b}{a} \in [-1, +1]$$

$$\frac{a \times_* b}{b} \in [-1, +1]$$

$$a(\underset{i \in U \cup \{a\}}{\Sigma_*} i - \underset{i \in U}{\Sigma_*} i) \geqslant 0$$

By using the analogy of matrix multiplication, we can define a multiplication operations between *rating* adjacency matrices of a *trust graph* which means we combine all the paths from i to j to evaluate the overall trust (*rating*) from i to j. The following equation means that each element $d_{i,j}$ in the resulting matrix is the combination of passing operations between elements $s_{i,k}$ and $t_{k,j}$ in adjacency matrix $\mathbf{S}$ and $\mathbf{T}$ where k is an intermediate identity document in $\mathcal{V}$.

$$\mathbf{S}_{(\mathcal{S}\times\mathcal{V})} \times \mathbf{T}_{(\mathcal{V}\times\mathcal{T})} = \left[\underset{k \in \mathcal{V}}{\Sigma_*} (s_{i,k} \times_* t_{k,j}) \right]_{(\mathcal{S}\times\mathcal{T})}$$

Then we have the definition of the adjacency matrix of a trust network $\mathcal{G}$ to a power n, here n means the how many hops the *rating evaluation function* travels through between nodes:

$$\mathbf{A}^n = \mathbf{A}^{n-1} \times \mathbf{A}$$

We use $\alpha_{i,j}$ to denote the entry in the (i,j)-th position of $\mathbf{A}^n$. Thus we have the following expression to solve the overall trust value from x to y with most n trust passing steps:

$$f_{\mathcal{G},n}^{\mathbf{RUT}}(x,y) = \alpha_{x,y}$$

We find that evaluating the trust (*rating*) on a graph is something like calculating the current in the circuit.The circuit is just a graph. So maybe Ohm's law[30], Kirchhoff's circuit laws[31] and other ways to calculate the current is useful. We can regard the *rating*s in the *trust graph* as the conductance[32] (reciprocal of resistance) (mapping domain of *rating* $[0,1]$ to domain of conductance $[0,+\infty)$) in the circuit. The passing and combining operations of the trust (*rating*) is just like electric current going through serial and parallel circuits. But the problem is the *trust graph* is a weighted directed graph, which can not regarded as the purely resistive AC circuit. Here we give a simple solution of *rating evaluation function* with most trust passing step 2 (because with the constraint of that most trust passing step is 2, the current direction is easily determined, the *trust graph* can be easily transformed to a purely resistive AC circuit) which means a lot of *rating* infomation will be ignored. Then follow the law of circuit, we have:

$$a \times_* b = a \times b$$

$$\sum_{\substack{*\\ i\in U}} i = e^{\frac{1}{\sum_{i\in U}\frac{1}{\log(i)}}}$$

$$f_{\mathcal{G},2}^{\mathbf{RUT},\mathbf{e}}(x,y) = e^{\frac{1}{\sum_{k\in\mathcal{V}}\frac{1}{\log(a_{x,k}\times a_{k,y})}}}$$

If we want to have the following property:

$$\mathbf{A}^{m+n} = \mathbf{A}^m \times \mathbf{A}^n$$

[30] https://en.wikipedia.org/wiki/Ohm%27s_law
[31] https://en.wikipedia.org/wiki/Kirchhoff%27s_circuit_laws
[32] https://en.wikipedia.org/wiki/Electrical_resistance_and_conductance

We have to add the constraints on $\times_*$ and Σ_* concerning the associative, commutative and distributive properties, (easily proved like power of matrix):

Associative Property

$$\begin{cases} \underset{k=1,2}{\Sigma_*} a_k = \underset{k=2,1}{\Sigma_*} a_k \\ a \times_* b = b \times_* a \end{cases}$$

Commutative Property

$$\begin{cases} \underset{i}{\Sigma_*} \underset{j}{\Sigma_*} a_{i,j} = \underset{j}{\Sigma_*} \underset{i}{\Sigma_*} a_{i,j} \\ (a \times_* b) \times_* c = a \times_* (b \times_* c) \end{cases}$$

Distributive Property

$$\begin{cases} x \times_* \underset{k}{\Sigma_*} a_k = \underset{k}{\Sigma_*} (x \times_* a_k) \\ (\underset{k}{\Sigma_*} a_k) \times_* x = \underset{k}{\Sigma_*} (a_k \times_* x) \end{cases}$$

Then we can have many solutions of $\times_*$ and Σ_*. The following two simplest examples are two of them.

$$\begin{cases} \times_* = \times \\ \Sigma_* = max \end{cases}$$

$$\begin{cases} \times_* = min \\ \Sigma_* = max \end{cases}$$

This result is unideal because it discards much information but it is just an example of a possible trust evaluation function that takes into consideration the aforementioned assumptions and goals. There may be many better solutions of trust evaluation functions and the optimal solution could vary under different scenarios. Trustor will customize their own trust evaluation functions based on their needs.

Subjectiveness

The evaluation of trust is subjective to each trustor, meaning that for the same trustee in the same context, different trustors may reach different trust evaluation results. For example, Bob's best friend Alice and Bob's employer might hold different degrees in trust in Bob's work ethic. NEO ID achieves subjectiveness

using the Trust Model by having each trustor build a local trust graph according to their trust profile to evaluate trust. Therefore, different trustors can have different trust graphs customized to their trust preferences and reach disparate trust evaluation values for the same trustee.

Proposal Model

The dominant Public Key Infrastructure (PKI) system for internet-based trust has mostly solved, albeit through a centralized method, the issue of who is who online. However, this system does not provide features to verify whether an entity is authorized for certain actions or possesses certain attributes. In the beginning of the internet when the number of users were small and many of them part of the same communities, knowing who someone equated to trusted knowledge of what that person could or could not do. This is obviously not the case anymore when billions of entity interact daily online and authentication is not enough to provide authorization.

Simplified PKI (SPKI)[33], provided ideas on how to achieve this authorization by proposing "structures bind either names or explicit authorizations to keys or other objects." Pretty Good Privacy (PGP) extends the features of PKI by separating verification claims into two categories: validity and authorization. Validity is akin to authentication, meaning that the entity is who it claims to be. Authorization is how much the entity can be trusted to verify other entities. Though PGP adds the authorization feature, it is only limited to authorizing authentications and not other abilities or attributes of an entity.

The *proposal model* of NEO ID outlines a design for identity documents to incorporate expressions that describe the set of attributes of the entity and the authorization scope of the entity. Our design achieves context-dependence for authorizations by restricting authorization scope to only capable verifiers. For example, an institution capable of accrediting higher-education degrees might not be trusted to provide proof of employment history. The *proposal model* also allows for the use of a rich set of expressions to describe any attribute or ability, verifiable through symbolic computation.

There are two types of proposals in NEO ID, attribute proposal that describes the set of attributes of the entity and scope proposal that describes the authorization scope of the entity. Scopes can be trasferred from one *identity document* to another by authorization but attributes cannot be transferred because they describe only the features of the $identity document* claiming those attributes. For Privacy reasons, we recommend that attribute proposal should be offline.

Attribute Proposal

Attributes describe the features, capabilities, and traits of an identity document. They are considered private information and are kept off-chain to ensure privacy.

[33] https://www.ietf.org/rfc/rfc2693.txt

Attributes become more vague and encompassing as they are passed to achieve selective disclosure. For example if Alice is 24 years old and she needs to present a proof to an American bar that she is over 21, she would be able to provide a verified claim that her age > 21 but does not have to reveal her exact age. Because it is difficult to remove claims to identity documents after they have been issued, it is recommended to include an expiration limit on the proposal's validity period.

$$x \in proposal_{\mathbf{attr,parent}} \rightarrow x \in proposal_{\mathbf{attr,child}}$$

For example, the following proposal may means a person with name 'Alice' and passport number 'E5443443'.

$$\bigcap \begin{cases} name \in \{\mathbf{Alice}\} \\ passport \in \{\mathbf{E5443443}\} \end{cases}$$

Scope Proposal

Scopes describe the ability of an identity document to authorize proposals of other identity documents (e.g. its domain of expertise). They are considered public information and are kept on-chain so the whole trust net can access them to create a local trust graph. Attributes become more specific and narrow as they are passed. For example, if an entity capable of authorization national-level information provides a claim to a city, the city can only issue city-evel claims and cannot issue claims on a national scale. Because it is difficult to remove claims to identity documents after they have been issued, it is recommended to include an expiration limit on the proposal's validity period.

$$x \in proposal_{\mathbf{scope,child}} \rightarrow x \in proposal_{\mathbf{scope,parent}}$$

The following proposal may means an authority who can only authorize a man in India with any name.

$$\bigcap \begin{cases} name \notin \phi \\ gender \in \{\mathbf{male}\} \\ address \in \mathbf{India} \end{cases}$$

Context-Dependence

Trust is context-dependent, meaning that a trustor can reach different trust evaluation values for a given trustee depending on the relevant context. For example, Bob's teacher Alice might hold full trust over Bob's educational background but has no information on his employment history. We achieve context-dependence in the Proposal Model through the use of scopes that restrict the authorization

abilites of *identity documents* so that they cannot issue claims in domains they have no expertise in.

Game Model

Operation and Permission

In the trust network, each role has different permissions for the *identity documents* and the *claims*. *Trustor* and *recommender* have the right to creat/remove/update the *claim* issued by itself(such as adjusting the *rating*) and the *identity documents* of their own. While *trustee* only has the right to creat/remove/update his own *identity document*.

Payoff of each Role

Trustor and *trustee* need to pay for the verification and *recommender* will earn the recommendation fee.

Punishment and Incentive

Recommender will make an effort to make the *ratings* point to him higher so he will attract more *trustors* and he will earn more. Improper behavior will decease the *ratings*.

As following pictures show, if RA increase the *rating* to TA improperly, then SA should decrese the *rating* to RA to punish him for his behavior.

If RB2 increase the *rating* to TB improperly, and RB1 did nothing to RB2, then SB should decrese the *rating* to RB1 to punish him for his inaction.

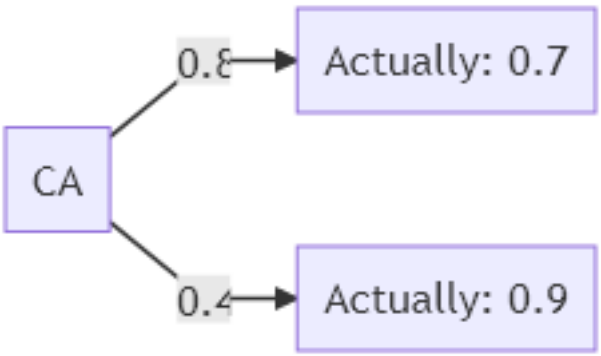
If RC2 increase the *rating* to TC improperly, and RC1 decrese the *rating* to RC2 to punish him, then SC should not change the *rating* to RC1 because RC1's behavior has balanced the trust network.

Evil

So what does evil mean? Let's discuss about the following two cases.

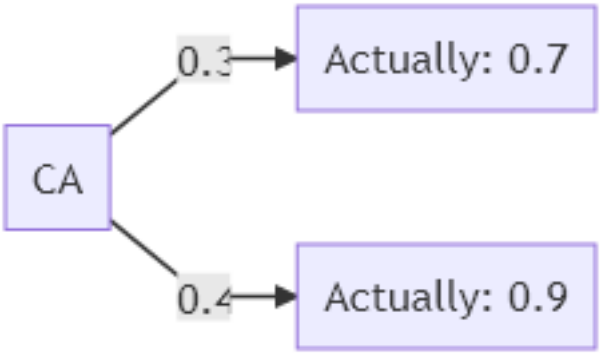
Out of order

You have a rating of 0.7,0.9 to A and B respectively while recommender CA give them 0.8 and 0.4, A has a higher score than B which is contrary to your opinion. This is a kind of evil. So if you are a *trustor* or *recommeder*, you should decrese the *rating* to CA.



Mapping in order

You have a rating of 0.7,0.9 to A and B respectively while recommender CA give them 0.3 and 0.4. This is not evil, maybe this CA is just more strict than you. And this kind of situation can be solved by mapping.



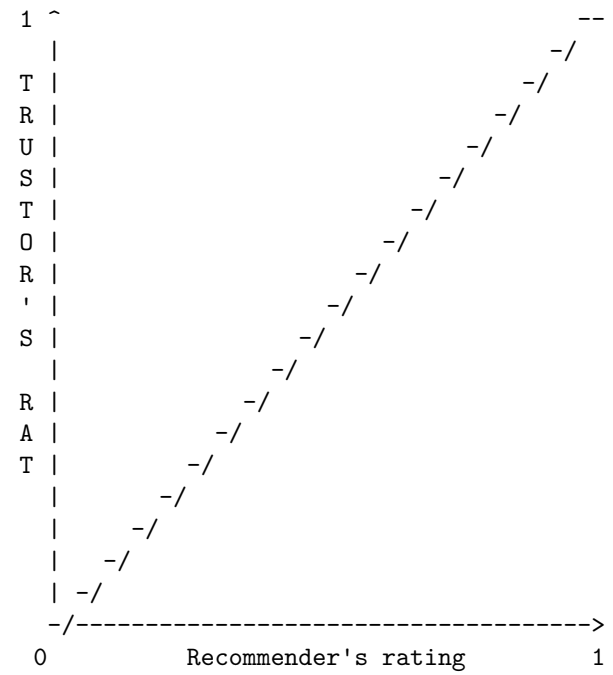
The Meaning of Rating

So actually the rating should be a mapping function $f : [-1, +1] \to [-1, +1]$ to map the recommender's *rating* to the trustor's *rating*. The functions must

remain monotonic. But a function is diffcult to represent on digital format, so we use a scalar *rating*, the slope of the two end point of the function, to represent the function, which means we recommend the mapping of rating to be linear. Because trust will gradually decay during the process of transition the maxium value of *rating* is +1. If you want to describe more details about your mapping function, you can add more info such as curvature and derivative in the *matadata* of the *claim*.

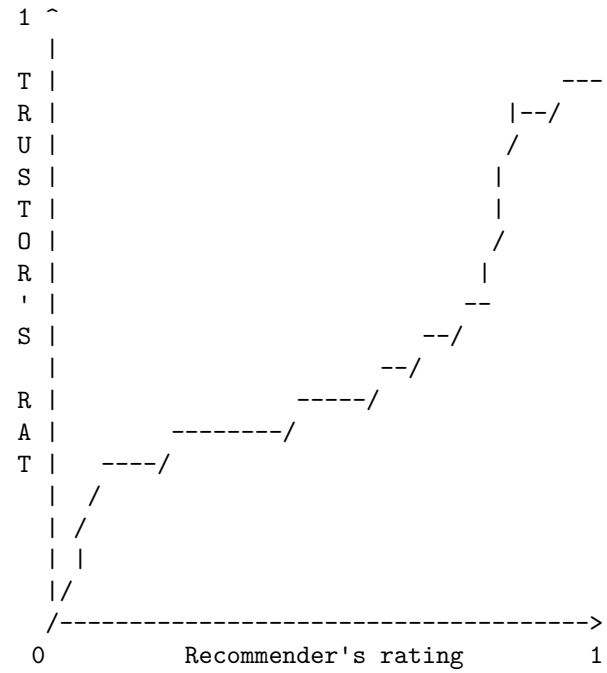
For example,

The mapping function in this picture is linear, and the *rating* is 1.

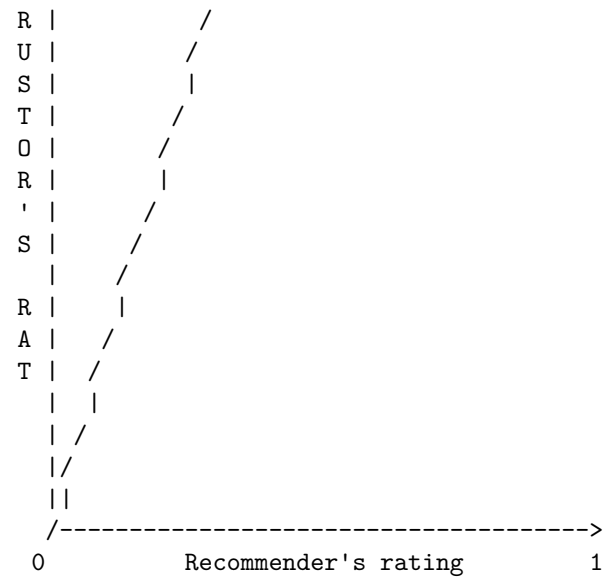


The mapping function in this picture is linear, and the *rating* is 0.66.

The mapping function in this picture is a curve, and the *rating* is 0.9, because the rating is the slope of start point $(0, 0)$ to the end point $(1, 0.9)$. You might add the function to *metadata* to describe an exact mapping.



Sometimes the standard of the recommender is more strict, then the slope may greater than 1. But the maxium rating is 1 which means the end point of the curve is $(1, 1)$. So in this case the *rating* is stiil 1. You might add the function to *metadata* to ensure an exact mapping.



Dynamism

The trust network will not be invariable all time. It will change over time and other factors, such as improper behavior, joining of a new *recommender*, exiting of a *recommender*.

Application

Centralized Management System

In today's world, the most commonly employed approach of identity authentication is based on centralized System like PKI. NEO ID solution can help to resolve the issues with traditional centralized System, Here we talk about PKI and KYC as examples. In these kind of scenarios, the rating will mostly be 1.

PKI

PKI system is s a Certificate Authority (CA) based system that adopts a centralized trust infrastructure. Communications over the internet are secured through the safe delivery of public keys and the corresponding private keys.Third-parties such as DNS registrars, X.509 Certificate Authorities (CAs), and social media companies are responsible for the creation and management of online identifiers and the secure communication between them. Unfortunately, this design has demonstrated serious usability and security shortcomings. For example, the PKI revocation mechanism is not efficiency; some companies spend significant resources fighting security breaches caused by misbehaving CAs; truly secure and user friendly communication still remains out of reach of for most netizens.

NEO ID solution can help to resolve the issues with PKI system through efficient mechanisms of certificate revocation and verification, eliminating single points of failure, and reacting fast to misuses of CAs. Blockchain is able to make the process in PKI more efficient, transparent, immutable, and avoid some attacks.

1. Eliminating a single point of failure

Traditional proposals try to solve the transparency issue in PKI by introducing one or more public logs. However, they are still subject to attacks or malfunction due to the trust placed in a centralized log operator. In order to prevent such problems, the trust should be distributed in such a way that no single log operator is able to control the log. In NEO ID, you could put the logs onto NEO blockchain to asure its transparency and security.

2. Revocation mechanism

Browsers have different implementations for revocation checking, introducing errors and uneven security. This is due to several factors, among which the lack of a high speed unified and trustless validator topping the list. But by managing the revocation status of certificates in the blockchain, revocation checking process can be simplified, unified and greatly sped-up – addressing both coding schemes and performance concerns of browser producers.

3. Improve verification efficiency

Certificate validation is a necessary process for each SSL/TLS connection, and must go through cumbersome trusted path construction and revocation checking phases. In the traditional PKI systems, certificate validation is completely done by browsers. However, if certificates are stored on the blockchain, trusted path construction can be done only once when they are being added to the blockchain, so that browsers can trust the certificates on the blockchain without further validation.

KYC

In this rapid development era, as the frequent changes of work, location and business services, KYC has become an indispensable part of many banks, insurance companies and other business activities. The current KYC standard process has greatly met the requirements of commercial and regulatory departments, but with the development of technology and the trend of policy, the process of KYC is becoming more and more complex and the cost is getting higher and higher. The cumbersome and inefficient KYC process has always been a pain point for banks and financial institutions in various countries, which consumes a lot of their financial resources.

NEO ID can help these enterprises establish trusted workflows to implement KYC processes. Allow participants to effectively use existing relationships and necessary data to create a secure “trust network” without revealing sensitive

data. In turn, this trust network can reduce operating costs and achieve a smoother and faster user experience.

Decentralized ID System

A decentralized ID system could be used to provide reputation service. And it can be extended from identifers for people to any entity. We can use DIDs to help us identify and manage objects, machines, or agents through their digital twins.In these kind of scenarios, the rating may not be assigned by user, it could be generated according connections between users, like follow on twitter.

reputation system

While the need for reputation services has been perhaps especially clear in the email world, where abuses are commonplace, other Internet services are coming under attack and may have a similar need. For instance, a reputation mechanism could be useful in rating the security of web sites, the quality of service of an Internet Service Provider (ISP), or an Application Service Provider (ASP). More generally, there are many different opportunities for use of reputation services, such as customer satisfaction at e-commerce sites, and even things unrelated to Internet protocols, such as plumbers, hotels, or books. Just as human beings traditionally rely on the recommendations of trusted parties in the physical world, so too they can be expected to make use of such reputation services in a variety of applications on the Internet.[34]

NEO ID allows users to customize trust evaluation models flexibly, including trust function and trust profile. Users can also specify the number of steps that trust could transfer, which is suitable for various application scenarios.

IoT

In the future, smart devices can replace people to deal with some daily work through the Intelligent Internet of Things(IoT). As the IoT matures, a lot of concerns are being raised about security, privacy and interoperability: first, security risks and privacy problems, the centralized service architecture gives vandals the possibility of inbreaking the network by attacking vulnerable links; secondly, higher operating costs while recording and storing the information of the devices will result in high storage and operating costs.

NEO ID can help to establish a digital identity for every networking device in order to realize identity binding and authentication, establishing a mutual trust mechanism between devices, and maintain consensus between devices without verification with the center. Due to its distributed components, the web scaled well beyond initial expectations, it can improve transparency and reduce the chain of trust, thus significantly improving the IoT security. When one or more nodes are compromised, the security of the whole network can still be maintained.

[34] https://tools.ietf.org/html/rfc7070

Conclusion

NEO ID is the firstly highly scalable system that transforms “trust or not” into “how much trust” with the newly proposed rating mechanism which allows users to customize trust evaluation models flexibly. Besides, The open standards on which NEO ID is based—DIDs and verifiable claims—represent a way for us to tap the unique cryptographic trust properties of a public blockchain. Most importantly, this new infrastructure can help us restore trust in the networks we rely on, power our global economy and connect our global society.