# Trusted-DID Specification

Christian Fries, Robin Lamberti

October 2019

## Contents

# 1 Requirements

1. **Create DID document:**
   The protocol must enable users to create a DID document, that complies to W3C DID Spec.
2. **Publish DID document:**
   The protocol must allow to publish a DID document on the tangle
3. **Add trusted IDs:**
   The protocol must allow users to add trust levels of published IDs to its list of trusted IDs
4. **Update Trusted IDs:**
   The protocol must allow users to update trust levels of published IDs to its list of trusted IDs
5. **Revoke trust of IDs:**
   The protocol must allow users to revoke trust of IDs by removing the trust level from its list of trusted IDs
6. **Query trusted IDs:**
   The protocol must allow users to query the list of trusted IDs from a known ID
7. **Query ID:**
   The protocol must allow users to query the DID document of an ID
8. **Query Trust of ID:**
   The protocol must allow users to query the trust of an ID from another ID's perspective
9. **Publish a claim:**
   The protocol must allow users to publish a claim for a certain ID
10. **Query claim:**
    The protocol must allow users to query a certain claim about an ID
11. **Publish attestation:**
    The protocol must allow users to attest a claim with its own ID
12. **Update attestation:**
    The protocol must allow users to update an attestation about a claim
13. **Revoke attestations:**
    The protocol must allow users to revoke an attestation made by it in the past
14. **Query trust of a claim:**
    The protocol must allow users to query the trust in a claim from ones ID perspective
15. **Query all claims of an ID:** The protocol could allow users to query all claims about an ID

# 2 Specification

## 2.1 Data model

All necessary data for the proposed protocol is stored on the IOTA tangle while making use of the IOTA specific design features such as data transactions and masked authenticated message (MAM) stream [1].
This ensures that all data is stored tamper proofed and provides a decentralized single source of truth.

### 2.1.1 DID document

The format of the DID document follows the W3C DID specification [2]. Listed here are the attributes of the DID document which are used by the protocol itself.

**Attributes:**

- **@context:** reference to the used DID specification

- **Method specific identifier:** unique identifier of the ID
- **Public key:** the public key of the DID's key pair

**Location:**

- **Address:** the Method specific identifier = root of MAM stream

#### 2.1.2 Trusted ID message

**Attributes:**

- **ID:** method specific identifier of the trusted party
- **Level:** trust value to the trusted party [0.0, 1.0]
- **Signature:** the signature of the trusting party (owner of the DID document)
- **Predecessor (optional):** The bundle hash of the previous list update

**Location:**

- **Address:** Second address of MAM stream

#### 2.1.3 Claim

**Attributes:**

- **Claim type:** A unique identifier for a specific type of claim, externally defined in a certain standard
- **Content:** the actual claim information
- **Claim target:** the method specific identifier of the identity that the claim is referring to
- **Claim issuer:** The method specific identifier of the claim issuer
- **Signature:** Signature of the claim issuer
- **Predecessor (optional):** The bundle hash of the previous version of the claim

**Location:**

- **Address:** Hash of the method specific identifier and the claim identifier

#### 2.1.4 Attestation Transaction

**Attributes:**

- **Claim:** Bundle hash of the claim transaction bundle
- **Trust level:** trust in the claim [0.0, 1.0]
- **Predecessor (optional):** Bundle hash of a previous attestation
- **Signature:** signature of the attesting identity

**Location:**

- **Address:** Hash of issuer and bundle hash of the claim concatenated

#### 2.1.5 ID Revocation

**Attributes:**

- **Id transaction:** Bundle hash of the did transaction bundle
- **Signature:** signature of the id itself

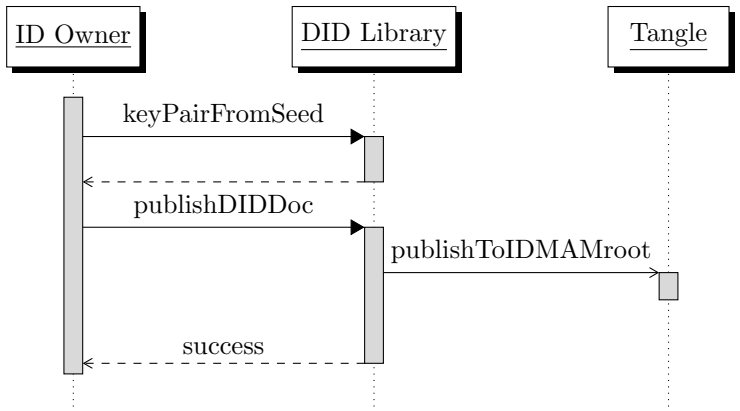
**Location:**

- **Address:** the Method specific identifier = root of MAM stream

## 2.2 Implementation of functionalities

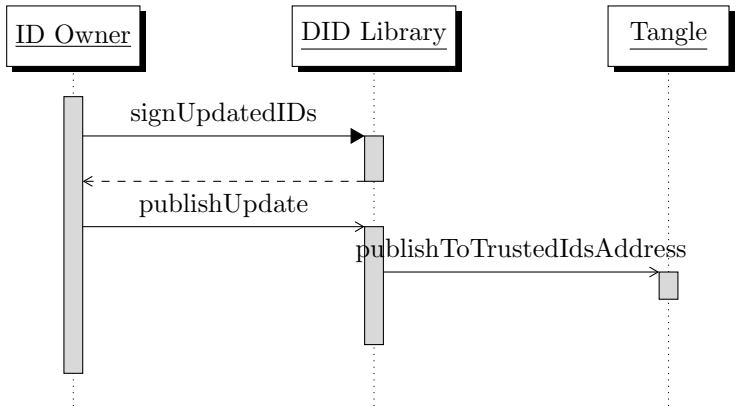
### 2.2.1 ID Creation

The ID creation starts with the generation of an IOTA seed, which consists out of 81 trytes (9,A-Z). Using this secret seed it is possible to generate an unique key pair which thereby is linked with the secret seed.
To publish the DID document containing the public key of the key pair, a masked authenticated message (MAM) stream [1] is created using the secret seed. Elliptic curve cryptography (ECC) is preferred over RSA due to the relatively short keys and signatures. In the specific implementation curve 25519 is used.
The DID document is published onto the root of the MAM stream as described in 2.1.1 which is from now own the method specific identifier of the newly created ID.
This procedure allows to link the key pair to the method specific identifier as only the secret seed owner can write into the MAM stream and publish a public key. Further it is advantageous that the DID owner just needs to store the secret seed securely as all other information can be obtained trough this. The described schema for publishing did documents was already implemented by [3].
During the use of the DID the key pair of the published public key will be used for signing.



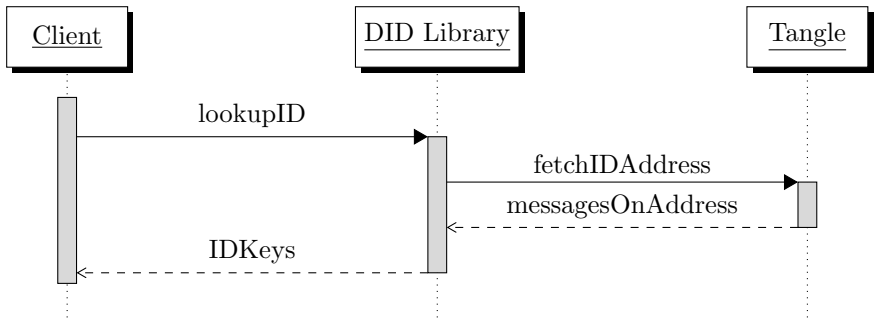

### 2.2.2 Update trusted ID's

The list of trusted ids is organized as incrementally build list which is stored on the address of first index of the id's MAM stream. The initial commit of the list on this address is signed by the public key of the identity and has no predecessor. Each further update links to the bundle hash of the transaction containing the predecessor update and gets signed by the id's key pair.



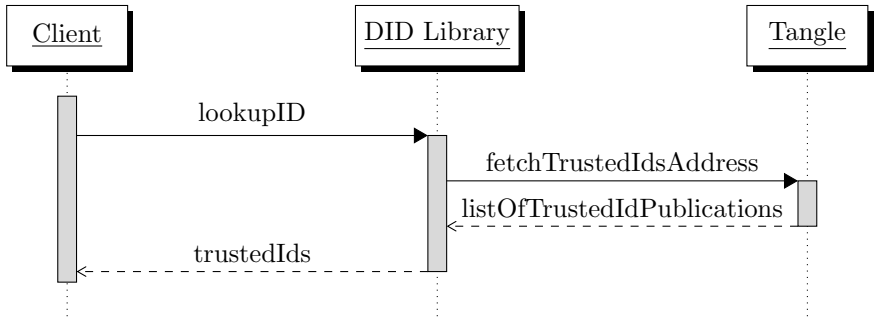

### 2.2.3 ID Lookup

As the ID (= method specific identifier) is equivalent to the MAM root of the ID, an ID lookup consists only of fetching the MAM root (address in the tangle) from the tangle to obtain the did document.




#### 2.2.4 Trusted IDs Lookup

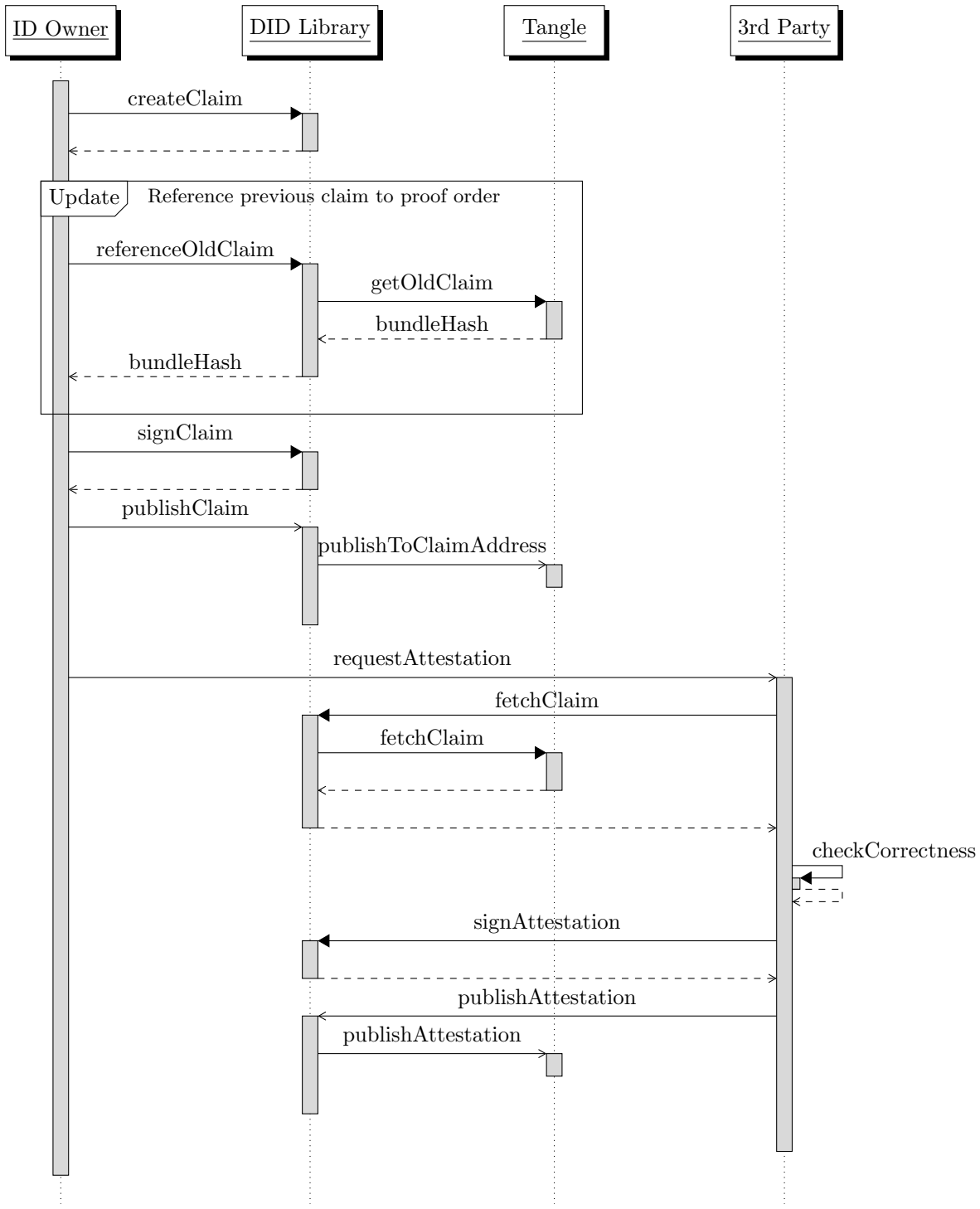
To fetch the list of all trusted id's of an id, the first index of the id's MAM has to be fetched. The index's address contains a linked list of incremental updates of the trusted id's list.
The linked list design, referencing the bundle hashes of the previous update transaction, ensures the order of the updates to guarantee the correct query of the current state of the list.




#### 2.2.5 Obtain/Update claim (ID properties)

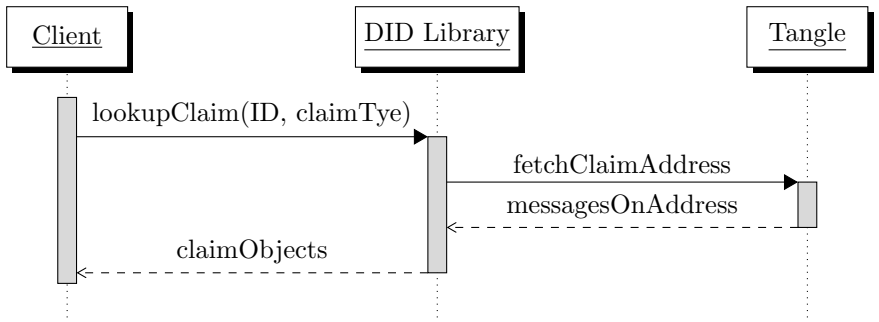
A claim about an id can be created by anyone, but has to be signed by issuer. In the sequence diagram beneath the claim issuer is the id owner himself. The claim is created following the format in 2.1.3, signed and published to the claim identifier specific address in the tangle.
If a certain claim type already exists, the new claim is considered to be an update and has to reference the bundle hash of its predecessor to maintain order.
However, a claim is worthless if nobody verifies it, regardless if it is an initial or an updating claim. Therefore an attestation of an desired party can be requested off-protocol. It is the attesting parties turn to take action and initiate the attestation of the claim. To do so, the attesting party fetches the claim in question from the tangle, presuming the affected id and the claim identifier is known, and checks it for correctness.
If this is the case the attesting party prepares the attestation message, signs and publishes it to the attestation address involved on the tangle where everybody can verify it.

**Example:**
A sensor manufacturer produces a sensor with a secure hardware element containing the secret seed. The product reveals its random identity to the producer and also publishes it in the tangle.
Now the producer or the device itself can create a claim about the devices identity's manufacturer on the address Hash(id + eClass:manufacturer) which contains the content 'Manufacturer plant Exampletown' for example.
After the claim was published it can be signed by all parties involved. In this case this might be the manufacturer, but also the device itself.



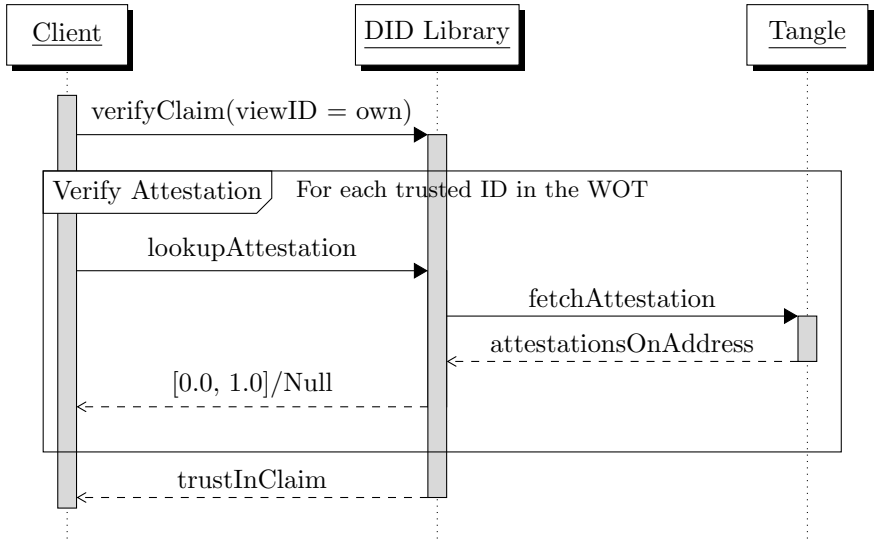

### 2.2.6 Claim Lookup

To lookup a claim the id about which the claim is, as well as the claim identifier has to be known to obtain the address where the claim should be written on (Hash(id + standard:type)).
Fetching the address on the tangle returns a list of all claims of the requested type about the id in chronological order.




### 2.2.7 Verify claim

For the process of verifying a claim there are two states of knowledge that can be presumed. The first is when the interrogator knows from which trusted id he wants to have a verification of the claim to have trust in it, the other one is when there is no knowledge about trusted ids at all.
In the first case the interrogator looks up the attestations made by the trusted id about the interrogated id. As each attestation carries a the unique bundle hash of the claim in question the attestation can be identified easily even if there are multiple attestations.
If the attestation of the claim carries the right signature the claim is considered correct.
In the second case the interrogator can make use of the protocol specific web of trust. Therefore he presumably takes his own id as a starting point and starts querying the attestations of all ids in his personal WOT about the interrogated id. If there are attestations about the interrogated id, the interrogator can calculate a subjective trust level about the attested claim with his preferred trust function.




### 2.2.8 Revocation of an attestation

Issuing a new attestation referring the same claim and referring the older attestation as predecessor with the trust level 0.0 is seen as a revocation.

### 2.2.9 Revocation of a claim

If the issuer of a claim revokes his attestation, the claim is considered revoked as well.

### 2.2.10 Revocation of identity

An identity can be considered revoked if all claims about are revoked as well or the identity revokes itself by publishing a signed message on the did address referencing the bundle hash of the published did document.

#### 2.2.11 Query trust in an identity

It is assumed that the personal WOT is already fetched from the Tangle and gets cached. If the desired identity is already in the personal WOT, the value can be read out directly. If not, the search depth can be increased until a link to the identity is found or the identity is assigned the value 0 for distrust. The exact calculation s explained in 3.

#### 2.2.12 Query trust in a claim

To obtain trust in a claim for each identity in the personal WOT it has to be checked, if an attestation exists on the tangle. Each attestation found is weighted dependent on the trust of the attesting party. For this process the same trust function as for identities can be applied.

# 3 WOT based trust

A major key of a decentralized trust protocol is to propose an adequate trust function to make trust quantifiable [4]. The objective is to design a trust function where trust is hard to gain and easy to loose. The trust range is defined in the interval [0,1] for trust expression and calculation. A value of 0 expresses distrust, whereas 1 is the highest achievable trust value. In other words: If a trustworthy part distrusts an identity, it can assigning the personal trust value of 0 to the identity. This distrust required an action whereby the distrust can be considered severe.
Another requirement of the trust function is that it should take into account the distances in the WOT with a reasonable measure to ensure that trust of a id more distant is taken into account less than one more close.
The protocol design provides the possibility to give an attestation or an trusted id a personal trust value. However, in the real world application it can be assumed that the input values will rather be binary. In this particularly case 1.0 or non existent rather than any other value. The trust function has to take that into account.
It has to be noted that the trust function is only required when establishing trust using a WOT (calculate subjective trust into an unknown id/claim) and not when trusting a single identity (verifying the manufacturer of a sensor).

## 3.1 Trust in an identity

### 3.1.1 Trust through claims

In the paper 'A Quantifiable Trust Model for Blockchain-based Identity Management' [4] the authors propose to derive the trust in an identity from the trust in the claims about the identity. This approach makes sense, as the identity's existence is consisting of claims about it.
However, for the protocol proposed in this paper there are several disadvantages using this approach. To do a trust calculation about an identity through its claims the interrogator demands a list of claim identifiers that need to be at hand to be considered. Having these in place, the interrogator now has to query all attestations of all trusted identities in his web of trust for each claim to come to an conclusion about the trust in the identity in question. This process is computationally expensive but can be reduced to specifying trust in identities in the WOT directly, as these are the values which determine the trust in the claim implicitly.
Further the protocol allows identities to carry a list of trusted identities which allows to query the trust in an identity directly.
Nevertheless, it has to be marked that the data structure of the protocol is generic enough to allow any kind of subjective trust calculation, including the trust through claims approach.

### 3.1.2 Trust through trusted identities

To obtain the trust in an unknown identity the interrogator queries the WOT from an initial stating id (most likely his own) to calculate the trust in the unknown identity based on the values obtained by the WOT query.
For the calculation of his subjective trust value the interrogator is free to choose any trust function that serves his needs.

## 3.2 Trust in a claim

Trust in a claim is established by the WOT of identities by evaluating attestations about the claim made by trusted identities in the WOT. The calculation of the trust value can be equivalent to calculating a trust value of an identity. The trust values of all attestations of trusted identities in the WOT are taken into account with the discount of the

personal trust value into the attesting identity. It may be advisable to additionally weight the opinions dependent on the attesting parties trust level.

## 3.3 Trust Function

An adequate trust function has to fulfil the following requirements:

- output a value in [0,1]
- take into account the distance (=depth in WOT) of the target
- weight the opinions dependent on the origins trust value
- consider that most values will be 1 or non existent

### 3.3.1 Naive Approach

The naive approach discounts each trust value with the origins trust value while taking into account the distance of the origin to the point of view as the denominator. This is done for the entire set of identities referencing the target identity on the current depth $d$. The naive approach can be described as the mean of the all identities in the WOT referencing the identity in question while discounting each value with the trust value in the identity referencing the queried identity and the depth in the WOT.

$$t_{s,t} = \frac{1}{|N_{t_o}|} \sum_{n=1}^{|N_{t_o}|} \frac{t_{(s,o_n)} * t_{(o_n,t)}}{d}, \text{ with } N_{t_o} = \{t_o \in T_o | t_o \text{ references } t_t\} \tag{1}$$

Note, that the calculation has to be done incrementally $\forall d \in D$ starting with $d = 1$.

Notation:

- $t_{s,t} = t_{self,target}$ = trust value of the target identity from the view point of self
- $t_{s,o_n} = t_{self,origin_n}$ = trust value of the origin identity n from the view point of self
- $t_{o_n,t} = t_{origin,target}$ = trust value of the target identity from the view point of the origin n
- $D$ = Depth: Distance between self and origin in the WOT

## 3.4 Trust Function Claim

When calculating trust in a claim, a claim can be treated just as an identity, including using the same trust function. Other than trust in an identity the trust is not expressed by carrying the claim in a list but by having dis-/trusted the specific claim with an attestation.

# References


[1] P. Handy, "Introducing Masked Authenticated Messaging," 2017. [Online]. Available: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

[2] D. L. Sabadello, C. Allen, D. Reed, M. Sporny, and Markus, "Decentralized Identifiers ({DIDs}) v1.0," W3C, Tech. Rep., 2019. [Online]. Available: https://www.w3.org/TR/did-core/

[3] L. BiiLabs Co. and Contributors, "Tangle ID," 2019. [Online]. Available: https://tangleid.github.io/

[4] A. Gruner, A. Muhle, T. Gayvoronskaya, and C. Meinel, "A Quantifiable Trust Model for Blockchain-Based Identity Management," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, no. September. IEEE, 7 2018, pp. 1475–1482. [Online]. Available: https://ieeexplore.ieee.org/document/8726703/