IPLD as a General Pattern for DID Documents and Verifiable Claims

a white paper from Rebooting the Web of Trust VII

By jonnycrunch, Anthony Ronning, Kim Duffy, and Christian Lundkvist

ABSTRACT

Since the emergence of the Decentralized Identifier (DID) specification at the Fall 2016 Rebooting the Web of Trust [1], numerous DID method specifications have appeared. Each DID method specification defines how to resolve a cryptographically-tied DID document given a method-specific identifier. In this paper, we describe a way to represent the DID document as a content-addressed Merkle Directed Acyclic Graph (DAG) using Interplanetary Linked Data (IPLD). This technique enables more cost-efficient, scaleable creation of DIDs and can be applied across different DID method specifications.

WHY IPLD

Content addressing through hashes has become a widely-used means of connecting data in distributed systems. IPLD is a way of representing hash-linked data to be used in content-addressed data retrieval systems such as IPFS. Other content that can be resolved using IPLD includes Git repositories and blockchains such as Bitcoin, Ethereum, and ZCash. IPLD enables creation of decentralized data-structures that are universally addressable, facilitating resolving content accross different protocols. It achieves this through an interoperable data model that represents various protocol formats. IPLD relies on Content Identifiers (CIDs) for content addressing. CIDs are a

Sponsors for the Rebooting the Web of Trust VII Design Workshop

self-describing, flexible, and interoperable way of expressing cryptographic hashes. It uses several multiformats to achieve flexible self-description, namely multihash for hashes, multicodec for data content types, and multibase to represent the base encoding of the CID itself. This interoperability makes IPLD a valuable structure for the DID document that can be used across a variety of DID methods or distributed ledgers and ensures cryptographic validity of the DID document.

In this paper, we demostrate how IPLD could be used as a general pattern for representing the DID document for numerous DID methods.

IPLD

IPLD uses an abstract model for linking data via cryptographic hashes, which enables link traversal to the referenced data via path **"/"** notation. This path notation has its roots in the Linux Filesystem Hierarchy Standard. A link in IPLD is represented in JSON as a "link object" and uses the path syntax **"/"** as the key to the object that is followed by the CID of the link.

For example:

```
{ "/" : "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq" }
```

This syntax for representing linked data can be expanded and used for other JSON structures. For instance, building on this example we can leverage JSON-LD semantics by inserting a **@context** to our IPLD object and apply the above as a link that resolves to the JSON-LD document that describes the attributes for our IPLD object.

```
{ "@context" :
      { "/" : "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq" }
}
```

JSON-LD is a syntax to serialize Linked Data in JSON and provide semantics without the overhead of a large rdf model [2]. Since both IPLD and JSON-LD are 100% compatible with JSON, the large number of JSON parsers and libraries are already available.

However, when using JSON-LD in a browser, it is impossible to discover the Base IRI after an http redirect (see #316), and the content of the `@context` can potentially change over time [3]. Finally, since a URI depends upon the security of DNS, DNS spoofing/DNS poisoning could offer a simple attack vector. Essentially, without much effort, an attacker can adjust the cache of a DNS server and begin pointing traffic from 'schema.org' (or any other desired host) to anywhere else on the internet or local LAN. Given the critical nature of the JSON-LD `@context` resource, the attacker can make a fraudulent signature pass as being valid.

Using IPLD, we can use the entire JSON data model and we can layer any JSON-LD on top of IPLD [4]. This will enable cryptographic guarantees to the authenticity of the JSON-LD schema and mitigate such an attack.

One additional point to emphasize is that the content loaded into IPLD is serialized using Concise Binary Object Representation (CBOR), allowing for deterministic representation and retrieval.

Content stored on IPLD is cryptographically retrievable regardless of the domain. The content can be retrieved via numerous http gateways including: ipfs.io, ipfs.infura.io, and cloudflare-ipfs.com, as well as a locally hosted gateway or via the command-line ipfs application.

Retrieval of IPLD content represented as dag-cbor from ipfs.io gateway and validating the content by reloading it into ipfs via command-line

```
> curl -s https://ipfs.io/api/v0/dag/get?
arg=zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq | ipfs dag put
```

outputs: **zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq**, thus validating the hash of the cid

Retrieval of IPLD content represented as dag-cbor from ipfs command-line

```
> ipfs dag get zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq
```

outputs the original JSON-LD schema link.

DID Document Creation and Updates using IPLD

At initialization of the DID document, the public/private keys pairs associated with the DID method specific identifier may or may not also have been generated. Additionally, the public/private key pair may or may not be the same as the master publishing key associated with the DID. In the example of the IPID DID method, the DID method specific identifier is the self-describing cryptographic multihash of the public key of an IPFS node and is different than the public/private key pair represented in the DID document. At present, RSA and Ed25519 elliptic curves are supported and there are plans to support secp256k1. Other DID methods that use IPLD to store the DID document may use alternative approaches to associate the DID document with the method specific identifier. For instance, in the muPort DID method, an Ethereum smart contract is used to point to the hash of the latest version of the DID document, and the document can be retrieved using the content-addressed IPFS system. The Sidetree DID method uses a scalable Merkle data structure that aggregates multiple DID document updates in a Merkle Tree and publishes the root hash in a blockchain such as Bitcoin or Ethereum. A sidetree node can use these data structures to aggregate the updates into a complete DID document

Since the DID document needs to contain information about where to discover the latest version of the DID document (which is a requirement of DID resolution), in IPID the `id` field corresponds to the DID method specific identifier (i.e. did:ipid:), but other DID methods specs can implement this in their own ways.

While the exact update details are up to the DID method, Example 1 shows how the resulting structure might appear using IPID, which publishes the hash digest of the DID document to IPNS.

Intrinsic to the integrity of any blockchain-based application is the existence of a `previous` field. In a blockchain data structure each block contains an ordered list of cyptographic transactions. A cryptographic reference to hash of the previous block is stored in the block header. Similarly, in this lightweight protocol based on IPLD, we have made use of the `previous` field with a cyptographic reference to the previous version of the DID document. This faciliatates traversal of the entire history of the DID document. One drawback to this approach is that this is a retrospective perspective. This highlights the importance of the `id` field, which allows an agent to `follow the tip` of the chain and at any point cryptographically resolve the latest version of the document.

In IPID, associating the DID document with a DID is accomplished by cryptographically publishing the CID to the IPNS public key associated with the identity owner (DID method specific identifier). Any updates to the DID document are saved to IPLD and the resulting hash is published to IPNS cryptographically associating the new CID with the DID (for IPID this is the multihash of the public key). IPID uses a PubSub model for realtime updates to the DID.

So far, on its own, this approach is not considered a comprehensive standalone blockchain solution. Most importantly: this does not faciliatate consensus of the document across peers, and timestamps are self attesting. The phrase "microledger" is often used to describe this approach. To overcome this shortcoming, the CID of the DID document can be anchored in a `proof of existence` smart contract (e.g. Truffle) created by the identity owner. Alternatively, cryptographic timestamping protocols such as openTimestamps (free) or Chainpoint could be utilized.

EXAMPLES

Proof of Existence Smart Contract to anchor CID of a DID document

```
pragma solidity ^0.4.23;

contract ProofOfExistence {

    event ProofCreated(
        bytes indexed cid,
        bytes did
    );

    struct Proof {
        bytes did;
        uint block;
    }

    address public owner;

    mapping (bytes => Proof) DIDbyCID;
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    constructor() public {
        owner = msg.sender;
    }

    function notarizeHash(bytes cid, bytes did) onlyOwner public {
        Proof proof
        proof.did = did
        proof.block = now
        DIDbyCID[cid] = proof;
        emit ProofCreated(cid, did);
    }

    function ProofRequest(bytes cid, bytes did) public view returns (Proof) {
        return DIDbyCID[cid];
    }
}
```

This smart contract has the added benefit of listening to updates to the IPLD DID document via Ethereum events.

Example 1: Demonstration of initialization of DID document

```
{
  "@context": {
    "/": "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq"
  },
  "created": "2018-12-01T03:00:00Z",
  "publicKey": [
    {
      "curve": "ed25519",
      "expires": "2019-12-01T03:00:00Z",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey"
    }
  ],
  "updated": "2018-12-01T03:00:01Z"
}
```

Example 1 shows the DID document before publication and before a cryptographic signature has been added. Note that the **id** and **signature** fields are omitted as it is not associated yet with the DID method specific identifier. Unlike other DID methods, with the DID document being represented as IPLD, we can directly link the **@context**, which is also represented on IPLD as a resolvable CID cryptographic link. Notably absent is the

previous field, as this is the genesis of the chain of objects that subsequent updates will reference (see below). This entire DID document when added to IPFS as IPLD has a CID of **zdpuAqiExr6k4AbWF6BuGkgUbVMZ7jbJyNvRz9z9yyRBxosPi** and will be used as the **previous** field in the subsequent updated DID document (Example 2).

Example 2: Intermediate DID document updated with id field associating this DID document with a DID method specific identifier for future resolution

```
{
  "@context": {
    "/": "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq"
  },
  "authentication": {
    "publicKey": [
      "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew"
    ],
    "type": "EdDsaSASignatureAuthentication2018"
  },
  "created": "2018-12-01T03:00:00Z",
  "id": "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew",
  "previous": {
    "/": "zdpuAqiExr6k4AbWF6BuGkgUbVMZ7jbJyNvRz9z9yyRBxosPi"
  },
  "proof": {
    "/": "z43AaGF42R2DXsU65bNnHRCypLPr9sg6D7CUws5raiqATVaB1jj"
  },
  "publicKey": [
    {
      "curve": "ed25519",
      "expires": "2019-12-01T03:00:00Z",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey"
    }
  ],
  "updated": "2018-12-01T03:00:02Z"
}
```

Example 2 shows the addition of the **id** field with the **previous** field linking to the hash of Example 1

Example 3: IPLD DID document updated with the addition of the signature field

```
{
  "@context": {
    "/": "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq"
  },
  "authentication": {
      "type": "EdDsaSASignatureAuthentication2018",
      "publicKey": [
          "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew"
      ]
  },
  "created": "2018-12-01T03:00:00Z" ,
  "id": "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew",
  "previous": {
    "/" : "zdpuAosm9NYNW5kG2h3SBoCZz5DYqyTgf6qopkxpih5cFhqmU"
  },
  "proof" : {
    "/" : "z43AaGF42R2DXsU65bNnHRCypLPr9sg6D7CUws5raiqATVaB1jj"
  },
  "publicKey": [
     {
      "curve": "ed25519",
      "expires": "2019-12-01T03:00:00Z",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey"
    }
  ],
  "signature": {
    "created": "2018-12-01T03:00:02Z",
    "creator": "zdpuAohuM1RWMpzwQgWz5jXFCdPtz1rhD82fuZBauUDuRzknt/publicKey/0",
    "message" : {
        "/" : "zdpuAohuM1RWMpzwQgWz5jXFCdPtz1rhD82fuZBauUDuRzknt"
    },
    "signatureValue":
"o9r6LxgoGN8FoaeeUA6EdDcv12GvDzFEmCgjWzvpur2YSQyA8W2r0SSWUK+nH5tMqzaFLun6wwZ1Eot37a
mGDg==",
    "type": "ed25519Signature2018"
  },
  "updated": "2018-12-01T03:00:00Z"
}
```

Example 3 shows the final DID document after it is associated with the DID method specific identifier and signature. In this case, it was published to IPNS using the IPID method spec. Note that **id** field is now populated and the updated document was pushed to IPLD, resulting in a CID for the final document of **zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP**(link). The chain of trust of the history of all edits can be done my simply following the **previous** link with the syntax **zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP/previous** and back to the original genesis of the document with a cid of **zdpuAqiExr6k4AbWF6BuGkgUbVMZ7jbJyNvRz9z9yyRBxosPi** with the syntax of **zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP/previous/previous**.

Similarly, with this syntax the `signature` field links to the publicKey of the CID natively without the need of a referenced `fragment` and the `message` (payload) that was signed. Additionally, a `proof` field has been added which is itself CID link that resolves to a `proof of existence` smart contract on the Ethereum blockchain that resolves natively or can be externally validated.

Retrieval of **`creator`** of the signature can be performed via any ipfs gateway

```
> curl -s https://ipfs.io/api/v0/dag/get?
arg=zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP/publicKey/0
```

output:

```
"curve":"ed25519","expires":"2019-12-
01T03:00:00Z","publicKeyBase64":"qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=
","type":"EdDsaPublicKey"}
```

Retrieval of the **`id`** of the **`creator`** can be performed via any ipfs gateway

```
> curl -s https://ipfs.io/api/v0/dag/get?
arg=zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP/id
```

output:

```
"did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew"
```

Example 4: IPLD DID document updated after key rotation

```
{
  "@context": {
    "/": "zdpuAmoZixxJjvosviGeYcqduzDhSwGV2bL6ZTTXo1hbEJHfq"
  },
  "authentication": {
    "type": "EdDsaSASignatureAuthentication2018",
    "publicKey": [
      "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew"
    ]
  },
  "created": "2018-12-01T03:00:00Z" ,
  "id": "did:ipid:12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew",
  "previous": {
    "/": "zdpuB1oR3vjYmkDc9ALfY7o6hSt1Hrg2ApXaYAFyiAW5E4NJP"
  },
  "proof": {
    "/": "z43AaGF42R2DXsU65bNnHRCypLPr9sg6D7CUws5raiqATVaB1jj"
  },
  "publicKey": [
    {
      "curve": "ed25519",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey",
      "status": "revoked"
    },
    {
      "curve": "ed25519",
      "expires": "2021-07-08T16:02:20Z",
      "publicKeyBase64": "jGp4OT1GktrZMdrkOM+zj8iE1IqCiqg2iH+rUZ93jhE=",
      "type": "EdDsaPublicKey"
    }
  ],
  "signature": {
    "created": "2018-12-01T03:00:04Z" ,
    "creator": "zdpuAt99xoa8i2BrcjwpY2H6ksXeaE28upionw2VuxRamBs6H/publicKey/1",
    "message": {
      "/": "zdpuAyvreXzQHqwv3rL8MaVPjNJjpLLa5Du3HcbpQL41XS35G"
    },
    "signatureValue":
"WDA3Dx7c+UWR37oglhkLNwxAbxXM4YbT7TpgmaCQ/rSqbtXgM3EpQ4mpkPXT5OBLH6bDai12Ank8SUHW47
JxCQ==",
    "type": "ed25519Signature2018"
  },
  "updated": "2018-12-01T03:00:04Z"
}
```

Example 4 shows an updated DID document that has revoked key/0 and created a new key/1, which is used for signing the new DID document. The `cid` for this final document is `zdpuAtrP6ZSDZj6izYQEbuUjuDRGHSa5L59BZDa1deRwAAZRQ` and is used to publish and associate it with the DID.

USE WITH PAIRWISE IDENTIFIERS

The IPLD pattern may also be a good fit to use for pairwise identifiers. A pairwise identifier is a DID that is meant to be used only with one other entity. The idea is that when setting up a pairwise DID you can do it by generating the DID document, and send the DID as well as the DID document to the counterparty. The counterparty can verify that the DID document hash is the DID. Then if the user decides to update the DID document (normally through the use of a digital signature from a "management key" specified in the DID document) they can just present the updated version of the DID document to the counterparty, who can then verify the signatures and store the updated document in a local database and not necessarily publish it publically.

Even though IPFS could be used for content addressing there would not be a need to connect to a wider IPFS network. The "ledger" in this case could just be a simple database hosted by each of the counterparties or a private `ledger` shared between them.

BENEFITS

One large advantage of the IPLD approach described here is that the identity owner does not need to use a blockchain when initially creating an identity, thus making creation of identities fast and low cost (if not free). The DID and DID document will be cryptographically coupled by hashing. Only when the identity owner needs to anchor their DID document will they need a more costly tool such as a blockchain. In addition, IPLD is perfectly suited for pairwise DID, when the creation and sharing of a DID document could be cryptographically generated and shared between only two parties and NOT saved to a public ledger. Each party can simply add the resulting CID to a local database.

Summary of the benefits include:

- Low cost / free scalable solution for DIDs
- Ability to use Bitcoin, OpenTimeStamps, ZCash, Ethereum and/or Ethereum smart contracts, or Github as an optional proof of existence
- Ability to use CBOR as a schemaless data model
- Cryptographic verifiable resolution of a document
- Ability to use microledgers
- Perfectly suited for pairwise DIDs
- Ability to self-host documents or pay through future services such as Filecoin
- Easily resolve previous versions of the DID by cryptographic linking to the previous CID
- Resolvable anywhere (local networks, Mars, online, etc.)

DRAWBACKS

- Not resolvable without hosting (This could be construed as a feature for pairwise DIDs)
- `@context` is not a reserved word in the IPLD specificatioin.
- `{ "/" : "<CID>"` is not currently valid syntax for JSON-LD

One additonal drawback to this approach results from the current DID draft specification, which designates **`publicKey`** as an array of objects; for example:

```
{
  ...
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }, {
    "id": "did:example:123456789abcdefghi#keys-2",
    "type": "Ed25519VerificationKey2018",
    "owner": "did:example:pqrstuvwxyz0987654321",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }, {
    "id": "did:example:123456789abcdefghi#keys-3",
    "type": "Secp256k1VerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyHex":
```

```
"02b97c30de767f084ce3080168ee293053ba33b235d7116a3263d29f1450936b71"
  }],
  ...
}
```

The current approach is to iterate over each of the keys in this array to identify the key included by reference. Specifically, the algorithm to use when processing a publicKey property in a DID Document is:

1. Let value be the data associated with the publicKey property and initialize result to null.
2. If value is an object, the key material is embedded. Set result to value.
3. If value is a string, the key is included by reference. Assume value is a URL.
    1. Dereference the URL and retrieve the publicKey properties associated with the URL (e.g., process the publicKey property at the top-level of the dereferenced document).
    2. Iterate through each public key object.
        1. If the id property of the object matches value, set result to the object.
4. If result does not contain at least the id, type, and owner properties as well as any mandatory public cryptographic material, as determined by the result's type property, throw an error.

A better approach may be to name the key by name within the object of **publicKey** rather than as an array; for example:

```
{
  ...
"publicKey": {
    "key-1" : {
      "curve": "ed25519",
      "publicKeyBase64": "qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg=",
      "type": "EdDsaPublicKey"
    },
    "key-2" : {
      "curve": "ed25519",
      "expires": "2021-07-08T16:02:20Z",
      "publicKeyBase64": "jGp4OT1GktrZMdrkOM+zj8iE1IqCiqg2iH+rUZ93jhE=",
      "type": "EdDsaPublicKey"
    }
  },  ...
}
```

In IPLD, this will allow for a more straightforward reference without a convoluted algorithm to accomplish this.

```
> curl -s https://ipfs.io/api/v0/dag/get?
arg=zdpuAu918t7r8bv2wvuJiasiq78oDCbZ62ecZCj43oBWvspzr/publicKey/key-1
```

with output:

```
{"curve":"ed25519","publicKeyBase64":"qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cE
K9mg=","type":"EdDsaPublicKey"}
```

This can be cryptographically linked back to the **`id`** with:

```
> curl -s https://ipfs.io/api/v0/dag/get?
arg=zdpuAu918t7r8bv2wvuJiasiq78oDCbZ62ecZCj43oBWvspzr/id
```

CONCLUSION

IPLD provides a robust framework for a lightweight microledger that cryptographically resolves the necessary data elements critical to the flow of distributed (not just decentralized) public key infrastructure. While this approach is already being used by IPID and muport we would encourage other DID methods to consider the merits of this approach.

DISCUSSION AND FUTURE WORK

In this brief paper, we have introduced the application of IPLD as a general pattern for representing the DID document and have highlighted its potential benefits and drawbacks and explained how it could be used across multiple DID method specifications. This technique enables a cost-effective and scalable solution for the creation and resolution of the data elements necessary to represent the cryptographic primitives to facilitate a distributed public key infrastructure. This model can be used to accelerate the adoption of truly self-sovereign digital identities.

In the future, we hope to formalize this approach with additional stakeholders and standard bodies. We will also pursue adding `@context` as a `SHOULD BE` reserved attribute for IPLD and `{"/" : "<CID>"}` as valid syntax for JSON-LD. In order for IPLD to be more broadly adopted as a way of representing the DID document we will need to updated parsers to traverse and resolve IPLD links

Finally, we look forward to the `proof of spacetime` that will be provided by Filecoin as an instrinsic anchoring mechanism that it will provide [5].

REFERENCES

[1] https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2016
[2] https://www.w3.org/TR/json-ld/
[3] https://github.com/json-ld/json-ld.org/issues/547
[4] https://github.com/ipfs/ipfs/issues/36
[5] https://filecoin.io/filecoin.pdf

ADDITIONAL CREDITS

Lead Author: *jonnycrunch*

Authors: *Anthony Ronning, Kim Duffy, and Christian Lundkvist*

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust VII design workshop. On September 26th through 28th, 2018, over 40 tech visionaries came together in Mississauga, Ontario to talk about the future of decentralized trust on the internet with the goal of writing 3-5 white papers and specs. This is one of them.

Leadership Team: Christopher Allen, Joe Andrieu, Kim Hamilton Duffy, Manu Sporny, and Heather Vescent

Gold Sponsors: Civic, Protocol Labs, Sovrin, an anonymous donor

Silver Sponsors: HTC, Microsoft, PTB Ventures, Spherity, Tierion

Community Sponsors: Blockchain Commons, Learning Machine, Legendary Requirements, Purple Tornado, Veres One

Workshop Credits: Christopher Allen (Founder), Joe Andrieu (Producer and Facilitator), Shannon Appelcline (Editor-in-chief), and Claire Rumore (Graphical Recorder)

Thanks to our other contributors and sponsors!

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot7/issues

The next Rebooting the Web of Trust design workshop is scheduled for the week of March 1st to March 3rd in Barcelona, Spain. If you'd like to be involved or would like to help sponsor the event, email:

rwot-leadership@googlegroups.com