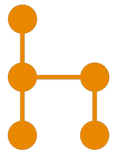
A DID (Decentralized Identifier) for Everything: Unified DIDs for Anything Decentralized

dDID
(derived-DID)
DADi
(Decentralized Autonomic Data item)

Samuel M. Smith Ph.D.
BlockChain Kickoff Summit 2019/01/29-30
sam@samuelsmith.org
https://github.com/SmithSamuelM/Papers



Early 2015 wanted to build credible portable decentralized reputation systems with data and algorithm provenance.

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/open-reputation-low-level-whitepaper.pdf

Needed decentralized identity (2016+)

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/Identity-System-Essentials.pdf

Which led to decentralized identifiers (DID) (W3C) (2016+)

https://w3c-ccg.github.io/did-spec/ https://w3c-ccg.github.io/did-primer/

Combined with zero-trust computing (2017+)

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/ManyCubed.pdf

Which led to decentralized autonomic data (DAD) (2018)

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/DecentralizedAutonomicData.pdf (RWOT6)

Which results in data flow chaining for data and algorithm provenance (2018)

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/A_DID_for_everything.pdf (RWOT7)

Decentralized Distributed Data Streaming Applications

Decentralized = controlled by multiple entities

Distributed = spread across multiple compute nodes

Maintain provenance (chain-of-custody) for distributed data under decentralized control undergoing various processing stages that follows perimeter-less diffuse trust (zero-trust) security principles.

DID = *UNIFIED* IDENTIFIER: 1

UUID: Universally Unique Identifier RFC 4122:

UUID type 1 -5

‘9866eb78-1376-11e9-bab5-58ef68134e82’

16 byte collision resistant decentralized identifier generated with random number generator and optional name spacing data.

Enables distributed applications to create unique identifiers without central authority

Prefixed name spacing allows for sorting and searching properties such as time order, lexical order, nesting etc.

DID = *UNIFIED* IDENTIFIER: 2

URI: Uniform Resource Identifier,

URI: Uniform Resource Locator,

URN: Uniform Resource Name

RFC 3986

```
scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]
```

Enables specifying derived resources from central root.
Mini language for performing operations on resources (ReST).

DID = *UNIFIED* IDENTIFIER: 3

Decentralized Self-Certifying Identifier:

Contains fingerprint of public member of cryptographic public/private key pair.

Key pair is generated by user not central registry.

http://www.sigops.org/ew-history/1998/papers/mazieres.ps

https://pdos.csail.mit.edu/~kaminsky/sfs-http.ps

Enables decentralized self-sovereignty over identifier namespace

Truly portable identifiers

DID = *UNIFIED* IDENTIFIER: 4

Hierarchically Deterministic Derived Self-Certifying Identifier:

selfcertroot:/path/to/related/data?derivation=parent/child/child/child

Enables low friction creation of identifiers on demand without having to store private keys

DID = *UNIFIED* IDENTIFIER: 5

Public lookup services for identifier(s) to find meta-data associated with identifier.

Resolvable identifier meta data.

Enables dynamic modification of identifier behavior and control

DID = *UNIFIED* IDENTIFIER: 6

Tupleizable (routable) Identifiers:

/channel/host/process/data = (channel, host, process, data)

Enables data flow routing overlay for distributed data processing systems.

DID = *UNIFIED* IDENTIFIER

*did:*method*:*idstring* (https://w3c-ccg.github.io/did-spec/)*

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?who=me
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue/my/stuff?name=sam#/foo/0
```

DDo = DID Document. Resolver lookup provides meta-data about DID.

Decentralized Identifiers Invert Compute Architectures

Conventional (centralized):

- Server creates identifiers (GUID, Database primary keys)
- Server timestamps
  - Event ordering relative to server
- Server manages keys,
  - AuthN/AuthZ is indirect via client to server proxy
- Perimeter security around servers
- Server is source of truth
  - Server controls changes/updates to resources
  - Signed at rest problematic
  - Encrypted at rest problematic
- Server's role is 2nd party in two party transactions between client to server to client.

Unconventional (decentralized):

- Client creates identifiers (DIDs)
- Client timestamps
  - Event ordering relative to client or vectorized relative to multiple clients
- Client manages keys
  - AuthN/AuthZ is direct peer-to-peer
- Perimeterless security
- Client is source of truth
  - Client controls changes/updates to resources
  - Server cannot make changes
  - Client signs at rest
  - Client encrypts at rest
- Server's role is either:
  - Trusted 3rd party in 3 (multi) party transactions between 2 (or more) clients and server
  - Agent or proxy for a client in two party transaction with another client.

DAD: Decentralized Autonomic Data

Decentralized: governance of the data may not reside with a single party, trust in the data provenance is diffuse, DID based.

Autonomic: self-managing or self-governing. Self-managing includes cryptographic techniques that make the data self-identifying, self-certifying, and self-securing.

Autonomic implies the use of cryptographic signatures to provide a root of trust for data integrity and to maintain that trust over transformation of that data

Key management is thus a first order property of DADi.

Reproduction, Rotation, and Recovery:

Pre-rotation & Hybrid recovery methods

Provenance for decentralized distributed data streaming including transformations

DADi: DAD item

DID, DDO, DADi, and dDID

DID = Decentralized Identifier

DDo = DID Document, provides meta-data about DID.

DADi = Decentralized Autonomic Data Item

dDID = derived DID = Unique DID format identifier derived from one *root* DID/DDo that provides meta-data for a large number of dDIDs

`did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue?chain=0/1`

`did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=`

Issues:

- Multiplicity of data items (DADis) and associated identifiers
- DDo lookup and caching may be expensive
- DID/DDo pair per DADi may not be practical so use dDIDs in DADis

dDID Management

dDID NameSpacing with HD-path: root + namespace + hd path

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue?chain=0/1
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:red?chain=0/1
```

dDID Sequencing: dDID + sequence number

```
did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/10057
```

dDID Database:

index key = anonymous dDID,

value = derivation from root DID

```
{

    "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=":
"did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?chain=0\1\2",

   ...

}
```

Example Signed DADi


```
{
    "id": "did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
    "data":
    {
        "name": "John Smith",
        "nation": "USA"
    }
}
\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Change Detection

Prevent replay attacks: either or both:

- sequence number in dDID
- *changed* field with monotonically increasing sequence number or date time


```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/10057",
    "changed" : "2000-01-01T00:00:00+00:00",
    "data":
    {
        "temp": 50,
        "time": "12:15:35"
    }
}
\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Zero-Trust Computing?

Never trust always verify

No such thing as *zero* trust

Really its *diffuse* trust

zero trust term used in 2013NIST report

Diffuse trust perimeter-less security model

Resources:

NIST: Developing a Framework to Improve Critical Infrastructure Cybersecurity 04/08/2013 Zero Trust Model for Information Security, Forrester Research.

http://csrc.nist.gov/cyberframework/rfi_comments/040813_forrester_research.pdf

https://www.nist.gov/cyberframework

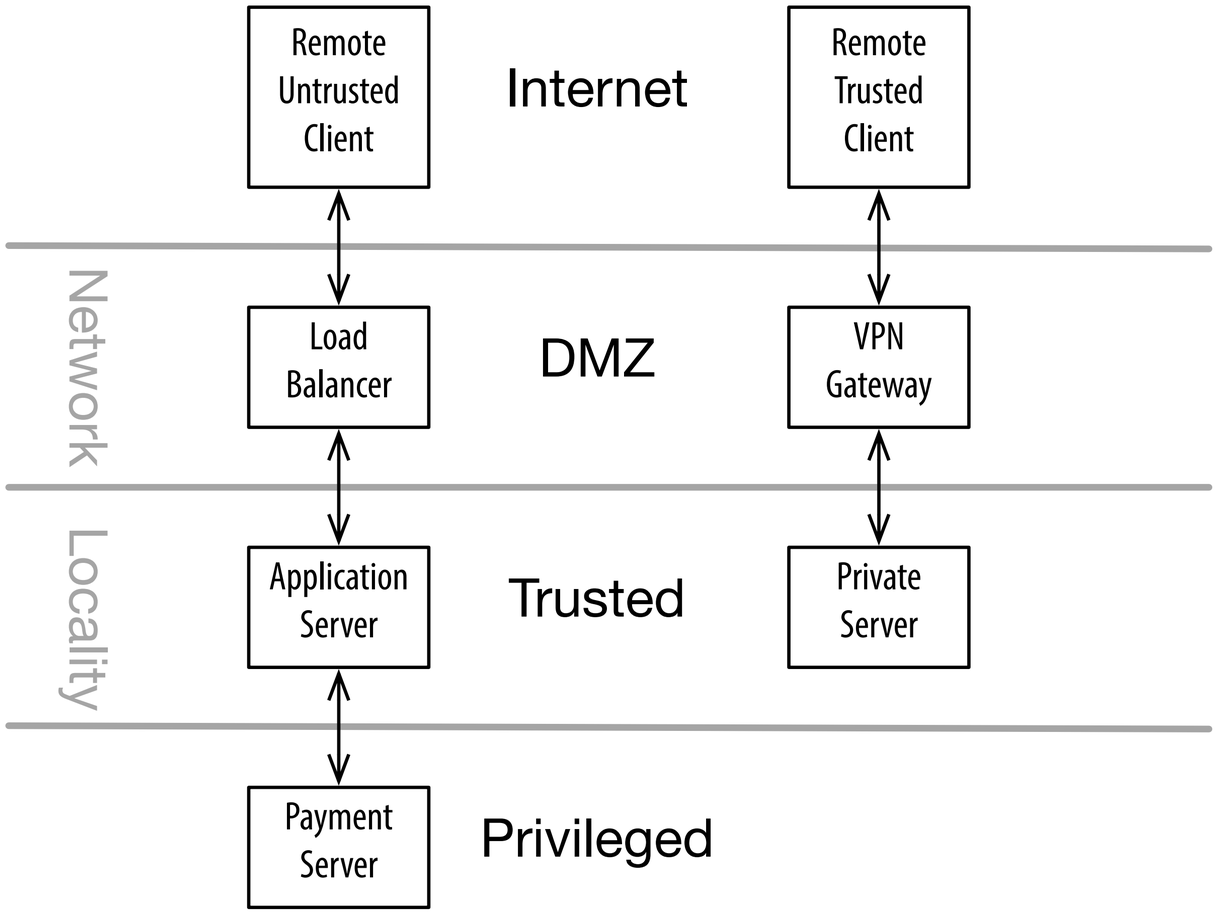
Zero Trust Networks 2017 Gilman & Barth

https://www.amazon.com/Zero-Trust-Networks-Building-Untrusted/dp/1491962194/ref=sr_1_1?s=books&ie=UTF8&qid=1499871379&sr=1-1&keywords=zero+trust+networks

Security Models

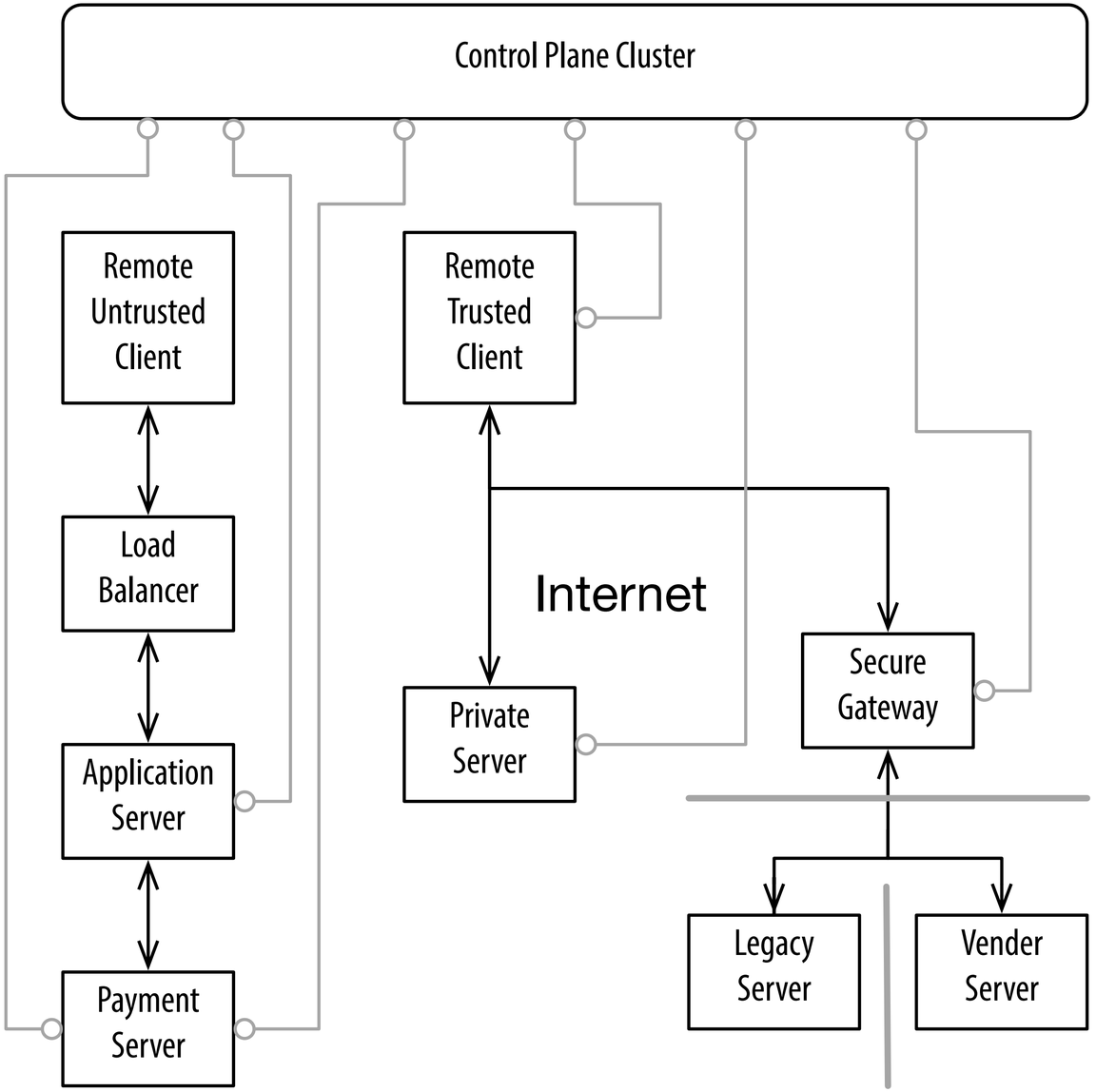
Locality Trust

Hard shells around Soft bodies



Zero Trust

Hard bodies everywhere



Diffuse trust perimeter-less security principles: 1

The network is always hostile both internally and externally.

Locality is not trustworthy.

Diffuse trust perimeter-less security principles: 2

By default, inter-host communication must be end-to-end signed/encrypted and data must be stored signed/encrypted using best practices cryptography.

Data is signed/encrypted in motion & at rest.

Diffuse trust perimeter-less security principles: 3

By default, every network interaction or data flow must be authenticated and authorized using best practices cryptography.

Verify every time for every thing.

Diffuse trust perimeter-less security principles: 4

Policies for authentication and authorization must be dynamically modified based on behavior (*reputation*).

Behavioral verification rules.

Diffuse trust perimeter-less security principles: 5

Policies must be governed by diffuse-trust distributed consensus.

Decentralized control.

Diffuse trust perimeter-less security principles: 6

By default, each data flow including all transformations must be end-to-end provenanced using decentralized identifiers (DIDs) and hence decentralized autonomic data items (DADis).

Dadify everything.

Diffuse trust perimeter-less security principles

Locality is not trustworthy.

Data is signed/encrypted/provenanced in motion & at rest.

Verify every time for every thing.

Behavioral verification rules.

Decentralized control.

Dadify everything.

Data Flow Provenance

Mechanism for tracing data item content and control (chain-of-custody) through a processing system including any transformations to the data item or its governance.

Includes flows with multiple sources and sinks of data, independently and in combination.

Includes verifying the end-to-end integrity of every data flow including any transformations (additions, deletions, modifications, and combinations).

An entity's influence on an application is solely based on the digital data flows that move between the entity and the other components of the distributed application.

These data flows are the entity's *projection* onto the distributed application.

If those projections consist of *DADIs* and every interaction of internal components consists of *DADIs* then we have a universal approach for implementing decentralized applications with total provenance of control and data within the application.

Chaining up DADi

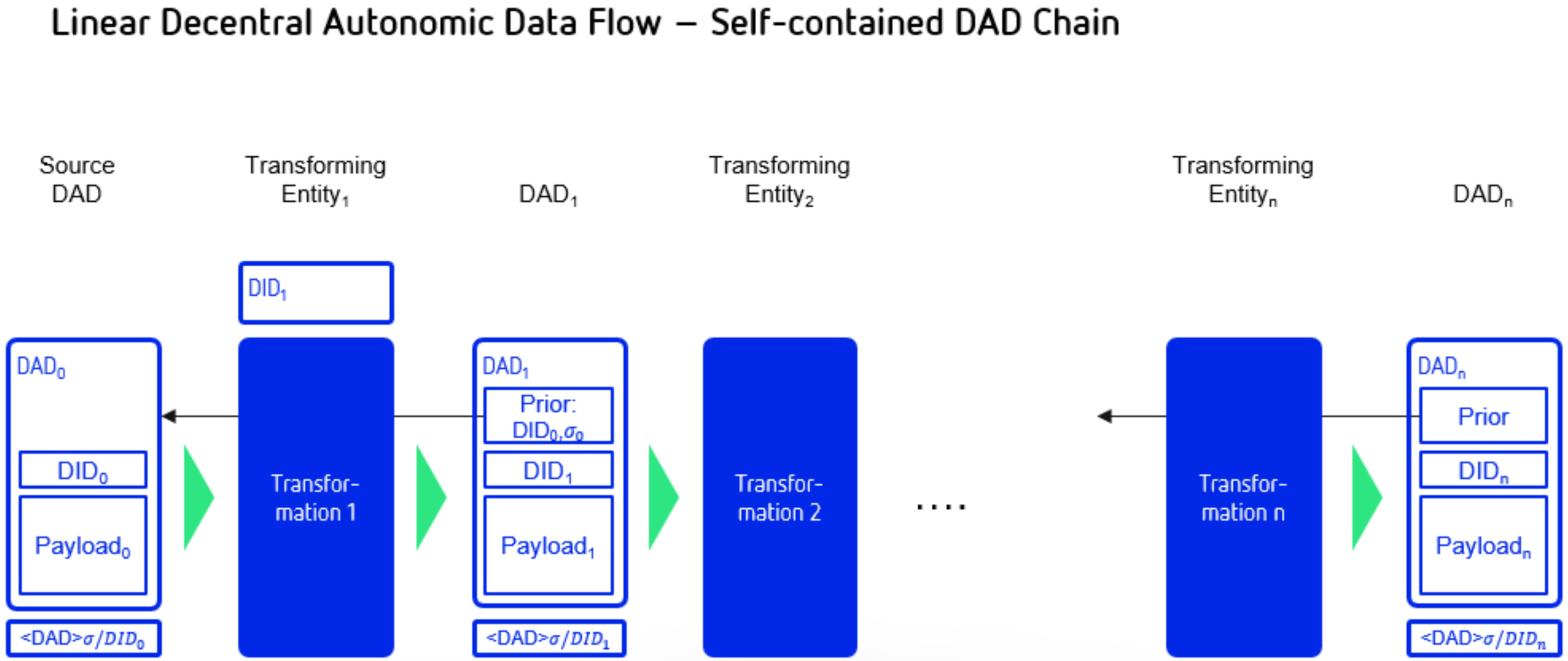
Self-contained virtual blockchain of the data.

IDs and signatures link transformation steps. (control and/or value)

Provides integrity and non-repudiation.

Use associated database to verify complete chain.

Chaining up DADi Diagram Linear



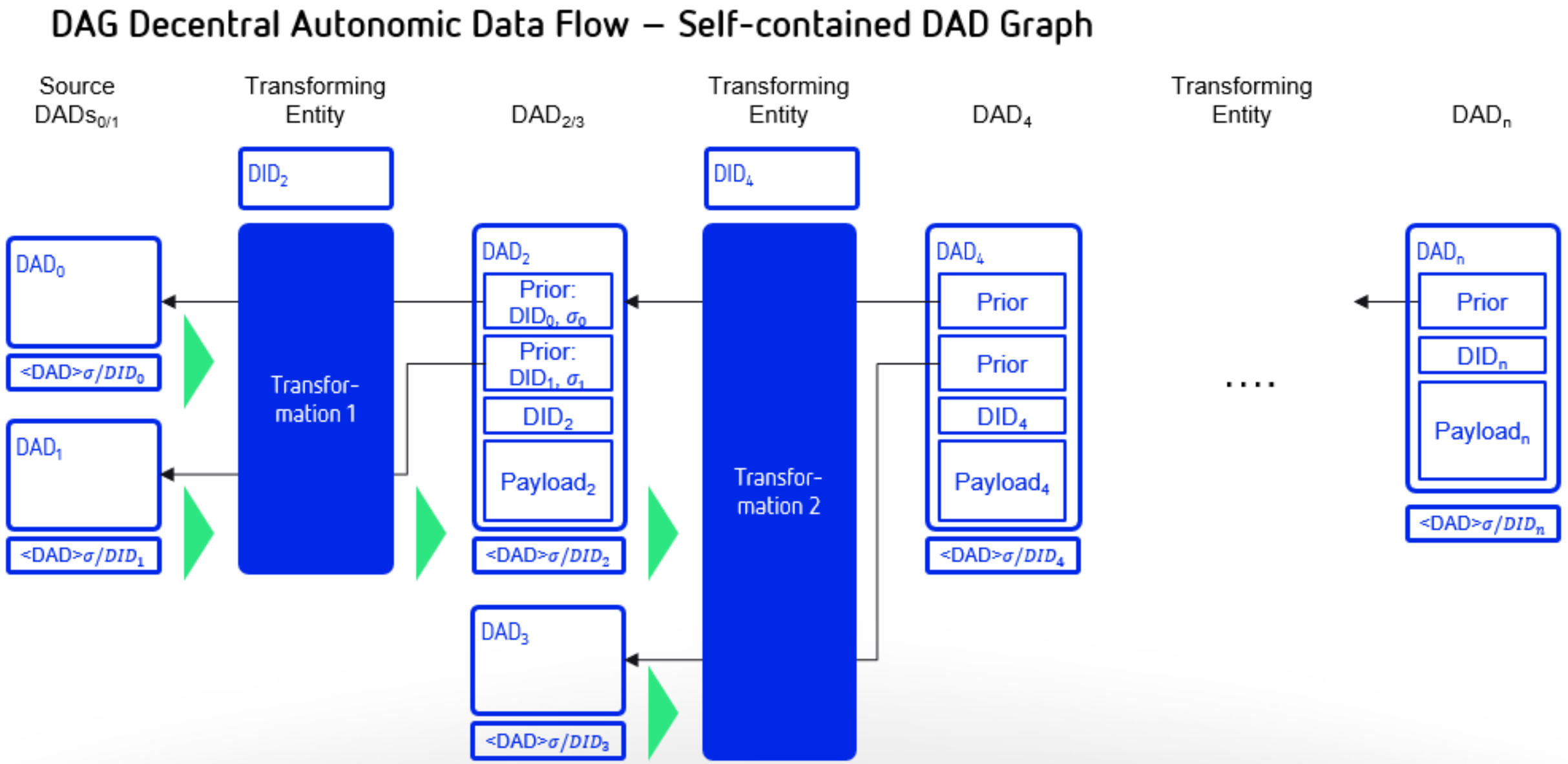
Chaining up DADi Example

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/alpha/10057",
    "changed" : "2000-01-01T00:00:00+00:00",
    "data":
    {
        "temp": 50,
        "time": "12:15:35"
    }
}\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

```
{
    "id": "did:dad:AbC7fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/beta/10057",
    "changed" : "2000-01-01T00:00:02+00:00",
    "data":
    {
        "temp": 50,
        "humid": 87,
        "time": "12:15:37"
           }
 "prior",
         {
                "id":  "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/alpha/10057",
               "sig": u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==

}\r\n\r\n
wbcj9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Chaining up DADi Diagram Multiplex



Chaining up DADi Example Multiplex


```
{
    "id": "did:dad:AbC7fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/gamma/10057",
    "changed" : "2000-01-01T00:00:03+00:00",
    "data":
    {
        "Avg temp": 55,
        "time": "12:15:39"
  }
      "priors",
         [
           {
                "id":  "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/alpha/10057",
               "sig":
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
      },
{
                "id":  "did:dad:WA27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/beta/10058",
               "sig":
j78j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
      },
]
}\r\n\r\n
dy3j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Conclusion & Discussion

sam@samuelsmith.org
https://github.com/SmithSamuelM/Papers
@SamuelMSmith

Entity

Something that has a distinct and independent existence either in the real or the digital world. Examples of an entity are:

Living Organism

Physical Object

Locations or Events

Machines and Devices in the Internet of Things (IoT)

Digital Asset, Data Set or Agent

Minimally Sufficient Means

Streaming data applications may impose significant performance demands on the processing of the associated data

Desire efficient mechanisms for providing the autonomic properties of DADis

Reproduction

Simple privacy via unique cryptonym (dDID) per pair-wise interaction context.

More sophisticated methods such as zero knowledge proofs may not be minimally sufficient.

dDIDs derived via some type of hierarchically deterministic algorithm allow for simple method to generated large numbers of public dDIDS without having to store the associated private keys. Only store the root private key

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?chain=0\1\2

did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=
```

dDID Re-Generation

On the fly DDIDs:

Data source is not identified so receiver generates DDID that is later correlated to or claimed by the data source

Public Derivation:

Client communicates with large number of public services

dDID is derived from root private key and public service identifier

Client does not need to store dDID but can re-derive on demand