DID for Everything
Unified Identifier

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

Samuel M. Smith Ph.D.
Internet Identity Workshop
2018.10.24
sam@samuelsmith.org

Background

Decentralized Autonomic Data and the Three R's of Key Management: RWOT Spring 2018

https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2018/blob/master/final-documents/DecentralizedAutonomicData.md

A DID for Everything: RWOT Fall 2018

RWOT Fall 2018https://github.com/WebOfTrustInfo/rwot7/blob/master/draft-documents/A_DID_for_everything.md

Motivation

Data streaming applications:

- Analytics and instrumentation for Web 3.0, Dapps, distributed data streaming, internet of things (IoT).

Maintain a provenance chain for data under decentralized control undergoing various processing stages that follows perimeter less diffuse trust security principles

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

"Zero" Trust Computing?

No such thing as *zero* trust

Really its *diffuse* trust

zero trust term used in 2013NIST report

Diffuse trust perimeter-less security

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

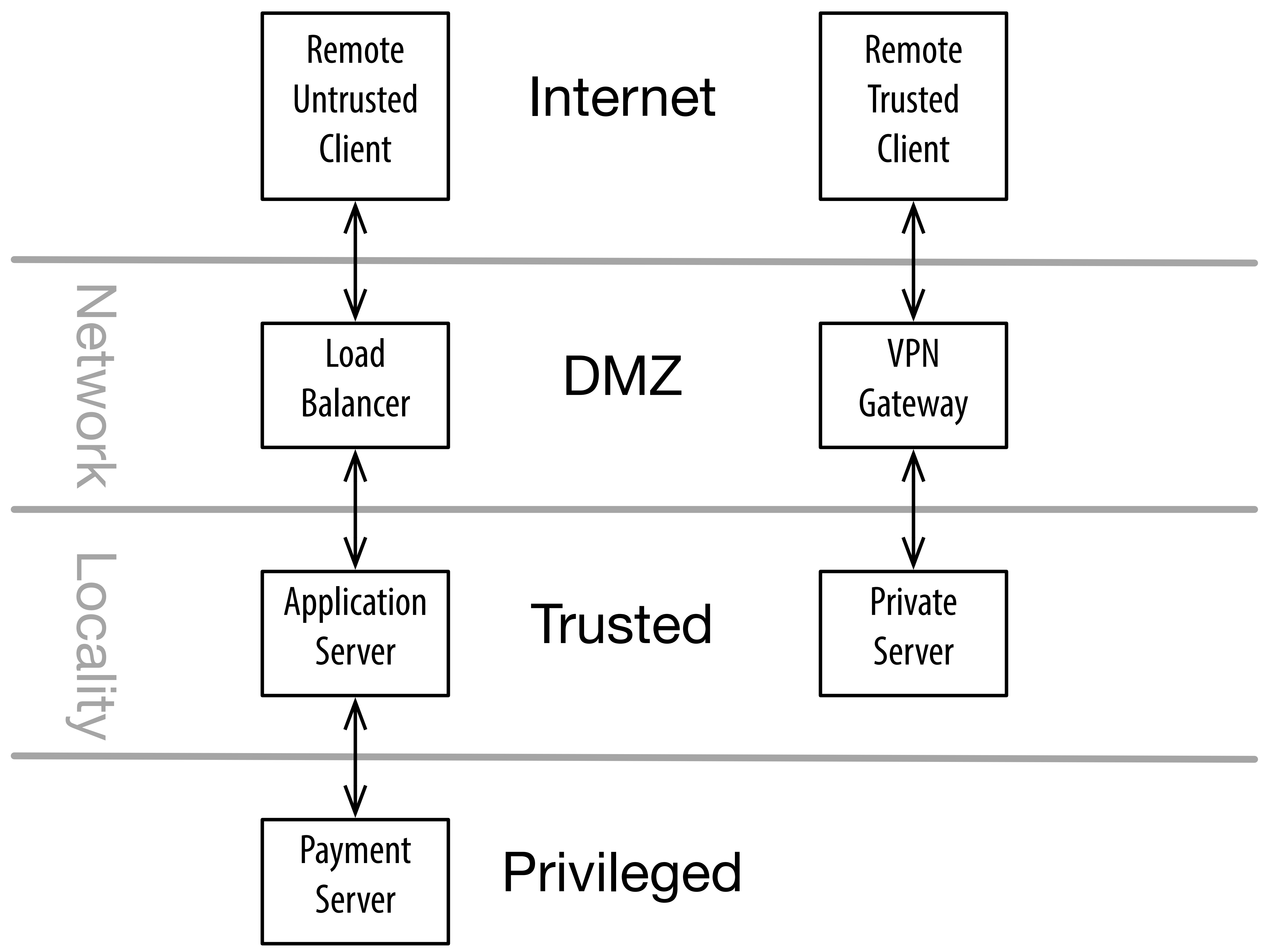
https://www.nist.gov/cyberframework

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

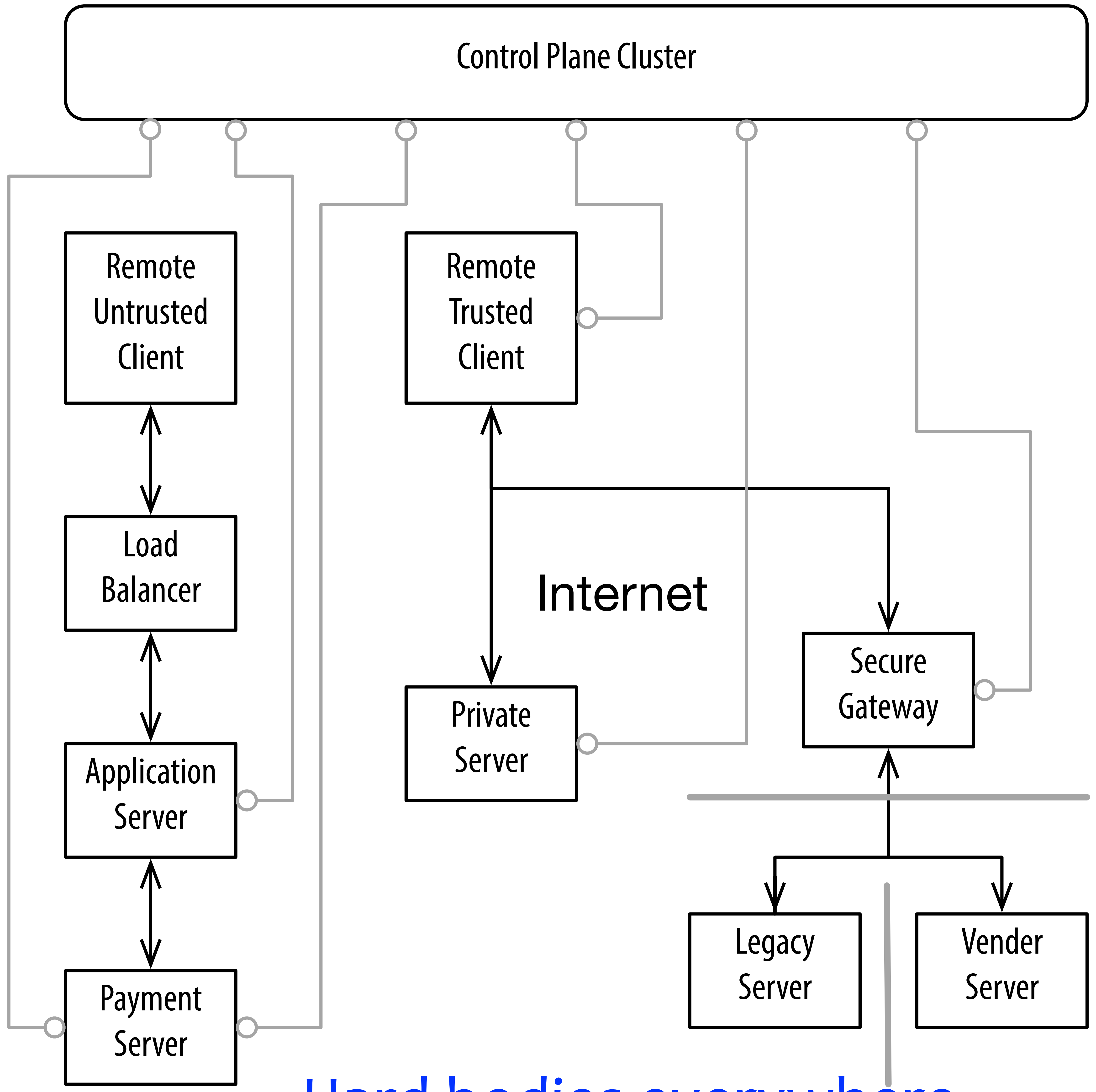
Security Models

Locality Trust



Hard shells around Soft bodies

Diffuse Trust



Hard bodies everywhere

Diffuse trust perimeter-less security principles

The network is always hostile both internally and externally; *locality is not trustworthy*.

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 at motion and at rest.*

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

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

Policies must be governed by distributed consensus. *Decentralized Control*

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).

DID = UNIFIED IDENTIFIER

UUID: Universally Unique Identifier RFC 4122: UUID type 1 -5

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 namespacing allows for sorting and searching properties such as time order, lexical order, nesting etc,

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.`

Decentralized Self-Certifying Identifier: Contains fingerprint of public member of cryptographic public/private key pair. public/private key pair is generated by user not central registry

Enables decentralized self-self-sovereignty over identifier namespace

Hierarchically Deterministic Derived Self-Certifiing Identifier:

selfcertroot:/path/to/related/data or parent/child/child/child

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

Tupleizable (routable) Identifiers: /channel/host/process/data = (channel, host, process, data)

Enables data flow routing overlay for data processing systems.

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
- 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
- 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

DADi: DAD item

Provenance for decentralized streaming data applications including transformations

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

A*utonomic:* self-managing or self-regulating. Self-managing includes cryptographic techniques for maintaining data provenance 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 first order property of DAD items.
- Reproduction, Rotation, and Recovery
- Pre-rotation & Hybrid recovery methods

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

DID, dDID, DADi

DID = Decentralized Identifier
https://w3c-ccg.github.io/did-spec/

```
did:*method*:*idstring*
```

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?who=me
```

DDO = DID Document, provides meta-data about DID
DAD Streaming = Multiplicity of data items and associated identifiers
DID/DDO pair per DAD item may not be practical
dDID = derived DID = Unique DID format identifier derived from one root DID/DDO that provides meta-data for a large number of dDIDs

Example Signed DADi

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

Data Flow Provenance

Mechanism for tracing data item content and control 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.

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 Management

dDID Database

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

dDID NameSpacing

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

dDID Sequencing

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

Change Detection

Prevent replay attacks:

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==
```

dDID 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

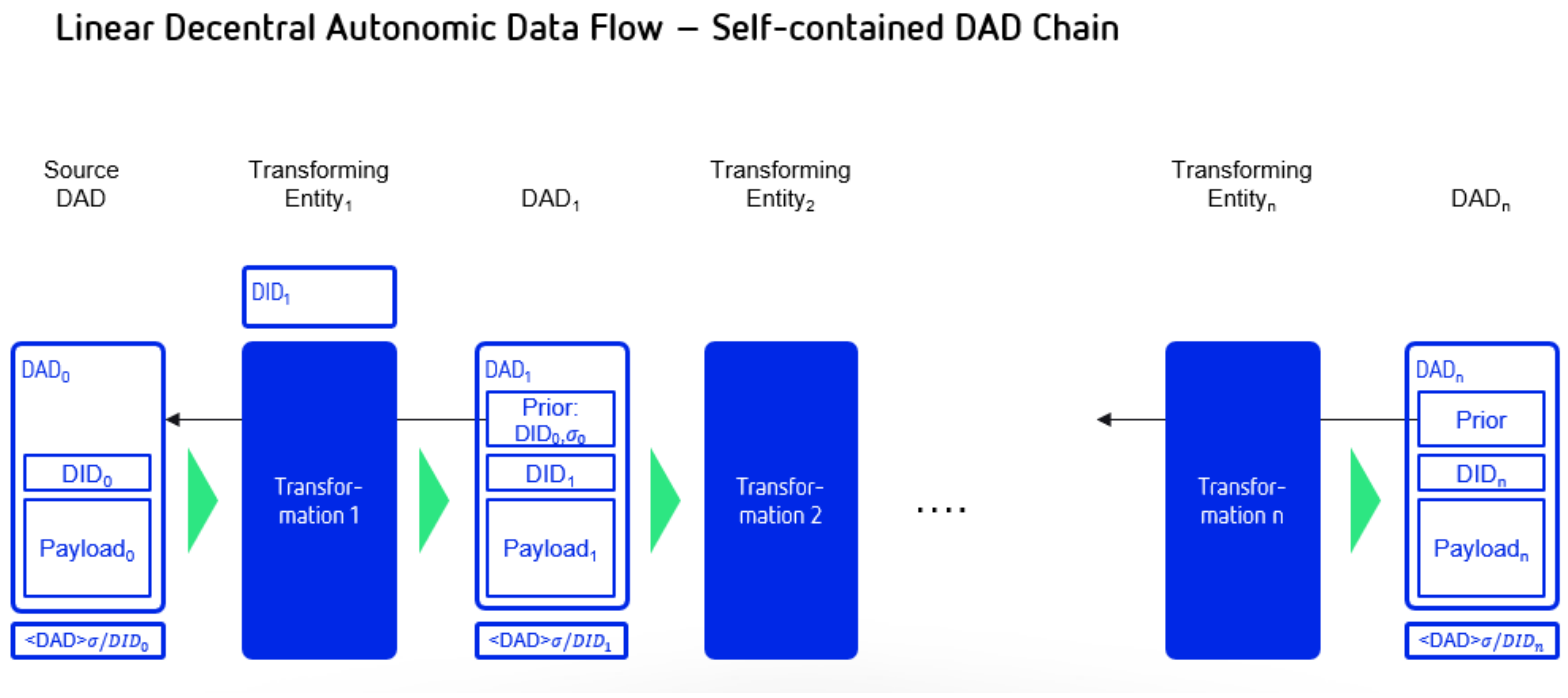
Chaining up DADi

Self Contained BlockChain of the Data with Signatures linking transformation steps.

Provides integrity and non-repudiation

Use associated database to verify complete chain.

Chaining up DADi Diagram Linear



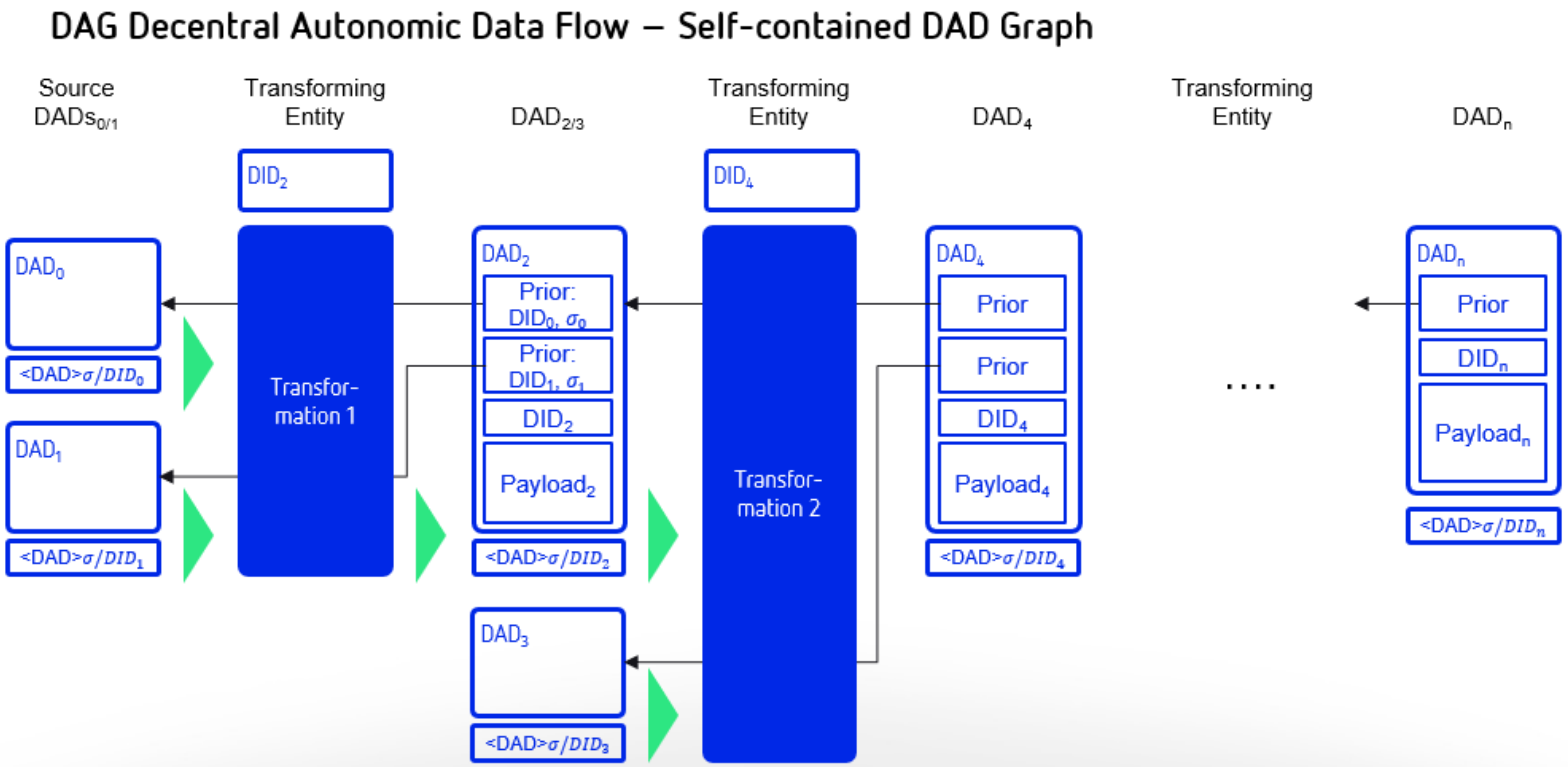
Chaining up DADi Example

```
{
    "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