MIRACL®

Trust Me or Trust Us?

4th January 2019

James Chapman
james.chapman@miracl.com

I’m James: Trust Me!

I’m James: Trust Me!

Really – says who?

Says Farah

I’m James: Trust Me!

Really – says who?

Says Farah

Who is she?

She’s my bank manager

I’m James: Trust Me!

Really – says who?

Says Farah

Who is she?

She’s my bank manager

How did she check it was really you?

Asked for my passport, 3 utility bills, and a quart of blood!

I’m James: Trust Me!

Really – says who?

Says Farah

Who is she?

She’s my bank manager

How did she check it was really you?

Asked for my passport, 3 utility bills, and a quart of blood!

But are you really the person Farah met?

Call her and I’ll speak to her – she’ll confirm it’s me

OK, I’ll allow you to buy me lunch ;-)

I'm James: Trust Me!

Really – says who?

Says Farah

Who is she?

She's my bank manager

How did she check it was really you?

Asked for my passport, 3 utility bills, and a quart of blood!

But are you really the person Farah met?

Call her and I'll speak to her – she'll confirm it's me

OK, I'll allow you to buy me lunch ;-)

Identity provider

Chains of trust

Verification and on-boarding

Authentication

Authorization

I'm James: Trust Me!

Trust me I'm an	Identity provider
	Chains of trust
Trust me I checked Trust the guy I checked with	Verification and on-boarding
Trust me with your credentials	Authentication
	Authorization

Passwords: A great way to lose your identity

Worst kind of distribution of **TRUST**. Repeated passwords means breach of one means breach of many

Who wants liability of providing Identity?

Horrible UX: too long to be useful or too short to be secure

E-mail	Bank	Social Media
Username1 Password1	Username2 Password2	Username1 Password1
Personal Information	Personal Information	Personal Information

Authentication

~Attestation

User loses control of personal information. Attestations stored by multiple parties which user has to **TRUST** to keep information safe

Password Manager

Using password manager just concentrates **TRUST** further

Password recovery can be less secure than initial on-boarding

Alternatives are also broken

- One Time Passwords
  - Seeds are stored at single point of **TRUST** that mustn't be lost [e.g. RSA and Lockheed Martin]
  - Open to social phishing and Man-in-the-middle
- Concerns over side-channel such as SMS
- Biometrics:
  - More like a user name than a password
  - Can be stolen
  - Not advisable (see NIST)
- 2-factor authentication is improvement but often still concentrates trust

Something I know
- Password
- PIN

Something I have
- OTP token
- Credit card

Something I am
- Fingerprint
- Face ID

The curse of public/private key

Certificate

- ID
- Public Key
- Signed

Rely on Certificate Authority (CA) – single point of **TRUST**
If CA compromised, system fails
Google lost **TRUST** in Symantec

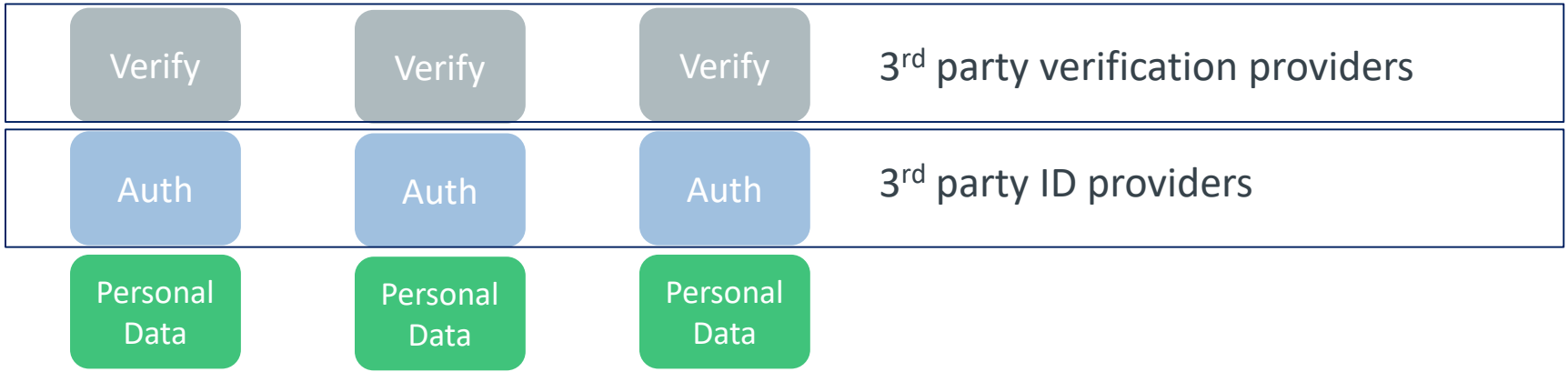
- Delivery of certificate/private keys is burdensome and does not scale to individual users
  - Work to scale public/private key on-going, including FIDO
- Need to **TRUST** storage for private key – forces user to carry extra HW or phone
  - Private Key is not distributed in any way

Trust me to:

- Keep your passwords
- Hold your personal data
- Sign your certificate
- Secure your private key

Can we distribute trust?

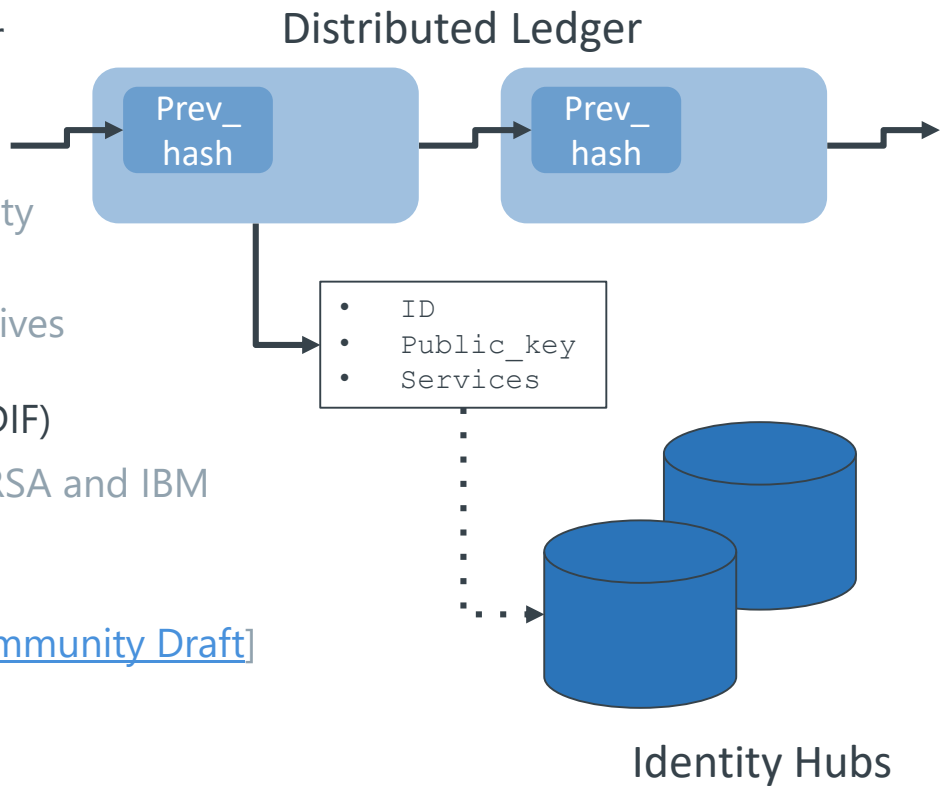
Moving away from vertical integration



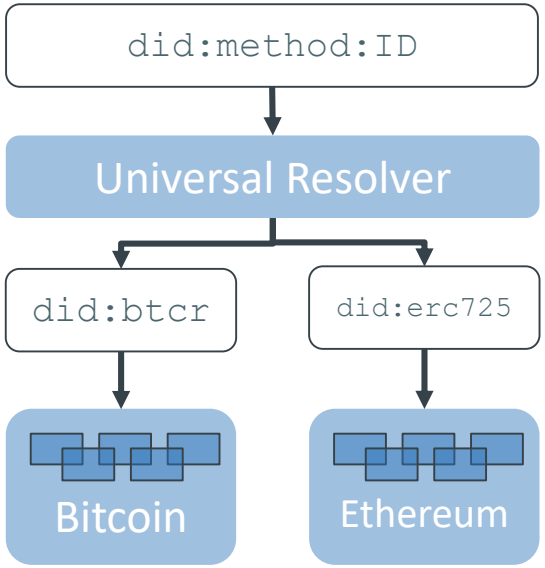
- Federation of Identity for authentication is critical and made possible by key protocols
  - Open ID Connect based on Oauth 2.0
  - SAML 2.0, often with single-sign-on for enterprise
  - ADFS, again for single-sign-on and used within Azure
- But still haven't really distributed trust, just moved it around

Move to Blockchain!

- ID:PublicKey guaranteed by Distributed Ledger
  - No need for Certificate Authority
- Identity information now with Entity
  - Not repeatedly stored at each relying party
- Trust in Entity comes from 'Web of Trust'
  - Built by attestations Entity receives and gives
  - Stored in Identity Hubs
- Driven by Decentralized Identity Foundation (DIF)
  - Significant backing including Microsoft, RSA and IBM
- Based around developing standards for:
  - Verifiable Credentials [W3C]
  - Decentralized Identifiers (DIDs) [W3C Community Draft]
  - Identity Hubs [on GitHub]



What DID you want to know?



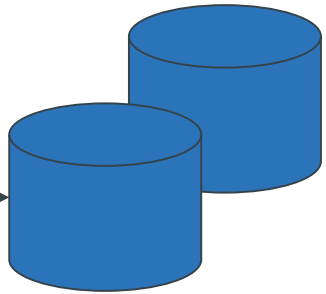
Persistent and immutable identifier.
One Entity can have multiple DIDs

Can use different DIDs for each relationship 'Pseudonymous DID'

Resolves the identifier to discover and retrieve information (a bit like DNS)

DID Doc

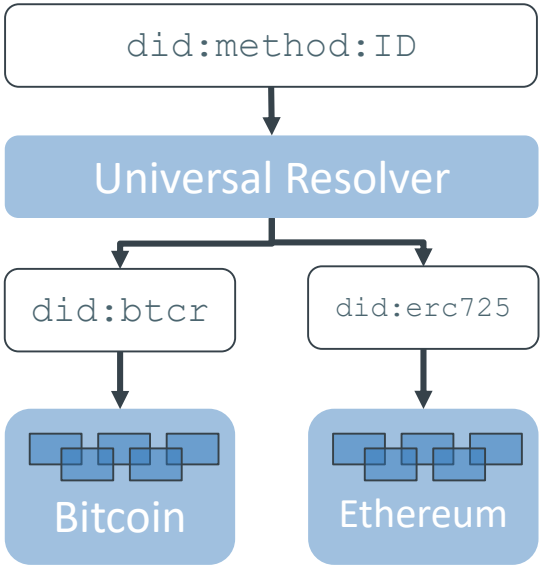
- ID
- Public Keys
- Service endpoints



Identity Hubs

Specific methods map onto specific Distributed Ledgers

What DID you want to know?



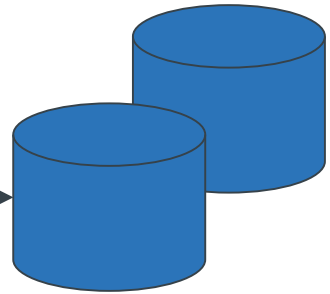
DID Doc written on Blockchain.
Similar to PKI certificate but no Certificate Authority
Provides service endpoints for gaining more attestations

Personally Identifiable Information kept separately in Entity Controlled Hub.

DID Doc is extensible

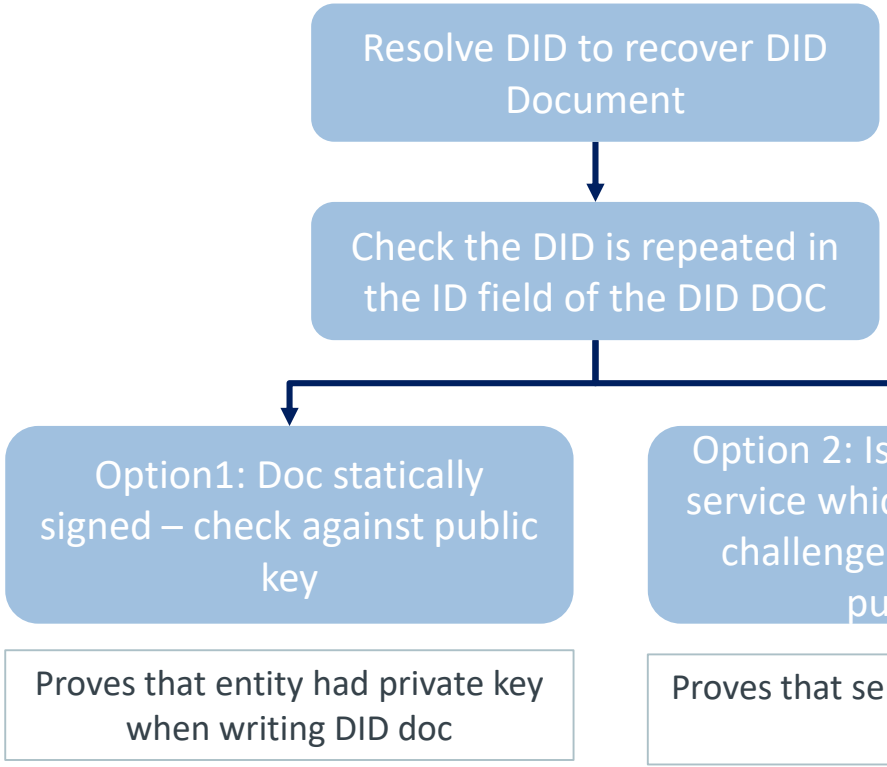
DID Doc

- ID
- Public Keys
- Service endpoints



Identity Hubs

How to authenticate the DID

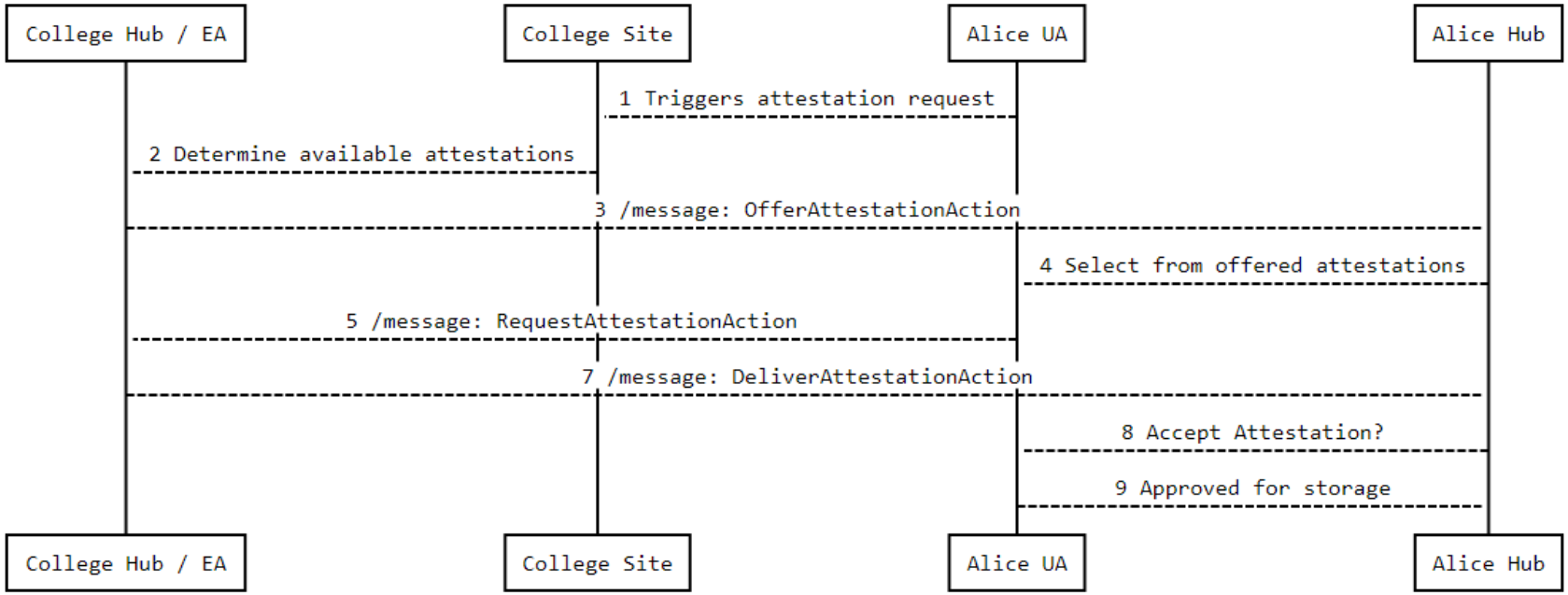


Further services used to yield more attestations about Entity:

- 2-factor authentication for OIDC token
- Proof of passport…

Entity needs to keep control of private key!

Will you vouch for me?



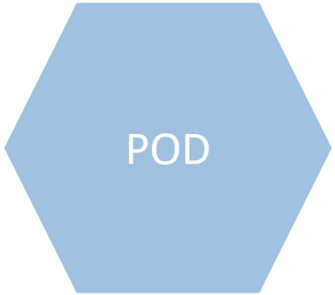
Taken from: https://github.com/decentralized-identity/identity-hub/blob/master/explainer.md

Why not control all your personal data?

SOLID (Social Linked Data) from Tim Berners Lee

- https://solid.mit.edu/
- https://www.inrupt.com/

All personal information kept in your POD



Every piece of data has it's own URL which can be linked:

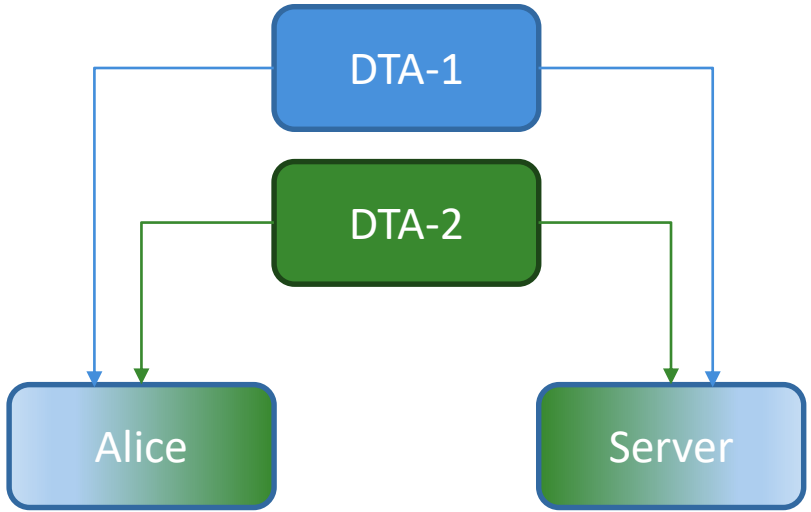
```
<https://mypod.solid/comments/36756>
 <http://www.w3.org/ns/oa#hasTarget>
  <https://yourpod.solid/photos/beach>.
```

Can Authenticate:

- WebID as identifier
- WebID-TLS – pub/priv key with certs on browser
- WebID-OIDC – using OIDC for delegated authenticaiton

ID-based encryption, no need for certificates!

- ID (e.g. e-mail) built into the cryptography
  - No need to bind them together with a certificate or on a blockchain
- Uses type-3 pairings on elliptic curves
- Essential feature is keys can come in pieces
- Distributed Trust Authority (DTA)
  - Two or more independent entities provide trust
  - Provision new identity with keys
  - Not needed after provisioning
  - Alice can't pretend to be server
  - Server can't pretend to be Alice



Finally we've solved the private key problem

- Split Alice's key into 2 parts
  - PIN (something Alice knows)
  - Token stored on device (something Alice has)
- Provides true 2-factor authentication
- Allows strong authentication without need for additional hardware

'Zero-knowledge Proof'
Alice proves her identity to authentication server without revealing anything useful

From ‘Trust Me’ to Trust Us’

- Many problems come from the ‘Trust Me’ approach
  - Keeping your passwords
  - Holding your personal data
  - Signing your certificate
  - Securing your private key
- Increasing moves to a ‘Trust Us’ model of distributed trust
  - Federated authentication (and 3rd party verification)
  - Using blockchains remove certificate authorities
  - Moving personal data back under control of person
    - Identity Hubs and PODs with API access
  - Modern cryptography allowing keys to be split and distributed

Thank You

James Chapman
James.chapman@miracl.com