Motivation
Motivation

Identity Provider Market:

Issues:

1. **Privacy** concerns:
   - Targeted advertisement, opinion shaping.
   - “Public safety”: Mass surveillance and data collection.


Issues:

1. **Privacy** concerns:
   - Targeted advertisement, opinion shaping.
   - “Public safety”: Mass surveillance and data collection.

2. **Liability** risks:
   - Data loss through leaks or hacks may result in existential legal implications (GDPR).
Issues:

1. **Privacy** concerns:
   - Targeted advertisement, opinion shaping.
   - “Public safety”: Mass surveillance and data collection.

2. **Liability** risks:
   - Data loss through leaks or hacks may result in existential legal implications (GDPR).

3. **Oligopoly**:
   - “There can be only one (two)”.
   - IdP market tends to degenerate.
   - Federation not widely used.
Primary objective: We must enable users to exercise their right to digital self-determination:
Primary objective: We must enable users to exercise their right to digital self-determination:

1. Avoid third party services for identity management and data sharing.
**Primary objective**: We must enable users to exercise their right to digital self-determination:

1. Avoid third party services for identity management and data sharing.
2. Open, free and decentralized service which is not under the control of a single organization, consortium or business.
Primary objective: We must enable users to exercise their right to digital self-determination:

1. Avoid third party services for identity management and data sharing.
2. Open, free and decentralized service which is not under the control of a single organization, consortium or business.
3. Free software.
**Primary objective:** We must enable users to exercise their right to digital self-determination:

1. Avoid third party services for identity management and data sharing.
2. Open, free and decentralized service which is not under the control of a single organization, consortium or business.
3. Free software.

⇒ Empower users to **reclaim** control over their digital identities.
What does an IdP do?

1. Identity provisioning and access control
   - Allow management of identities and personal data.
   - Facilitate sharing of identity data with third parties.
   - Provide up-to-date information accessible even if user is offline.
   - Enforce authorization decisions of user.
What does an IdP do?

1. Identity provisioning and access control
   - Allow management of identities and personal data.
   - Facilitate sharing of identity data with third parties.
   - Provide up-to-date information accessible even if user is offline.
   - Enforce authorization decisions of user.

2. Identity information verification and attestation:
   - “this is Alice’s email address”: Email provider.
   - “this person is living in Germany”: Sovereign state.

*We will revisit this further on.*
What does an IdP do?

1. Identity provisioning and access control
   - Allow management of identities and personal data.
   - Facilitate sharing of identity data with third parties.
   - Provide up-to-date information accessible even if user is offline.
   - Enforce authorization decisions of user.
   ⇒ re:claimID

2. Identity information verification and attestation:
   - “this is Alice’s email address”: Email provider.
   - “this person is living in Germany”: Sovereign state.
   ⇒ Not our department!*

*We will revisit this further on.
Introducing re:claimID
• re:claimID is a **self-sovereign** personal data sharing system.

• Other self-sovereign identity systems you may have heard about:
  • Sovrin (Hyperledger)
  • uPort (Ethereum)
  • NameID (Namecoin)
• re:claimID is a **self-sovereign** personal data sharing system.

• Other self-sovereign identity systems you may have heard about:
  - Sovrin (Hyperledger) ⇐ **Permissioned blockchain**
  - uPort (Ethereum)
  - NameID (Namecoin)
• re:claimID is a **self-sovereign** personal data sharing system.

• Other self-sovereign identity systems you may have head about:
  • Sovrin (Hyperledger) ⇐ **Permissioned blockchain**
  • uPort (Ethereum) ⇐ **Data shared off-chain: If user is offline data not accessible.**
  • NameID (Namecoin)
- re:claimID is a **self-sovereign** personal data sharing system.
- Other self-sovereign identity systems you may have heard about:
  - Sovrin (Hyperledger) $\Leftarrow$ **Permissioned blockchain**
  - uPort (Ethereum) $\Leftarrow$ **Data shared off-chain: If user is offline data not accessible**.
  - NameID (Namecoin) $\Leftarrow$ **Access control through central server (wat?)**
• re:claimID is a **self-sovereign** personal data sharing system.
• Other self-sovereign identity systems you may have head about:
  • Sovrin (Hyperledger) ⇐ **Permissioned blockchain**
  • uPort (Ethereum) ⇐ **Data shared off-chain: If user is offline data not accessible.**
  • NameID (Namecoin) ⇐ **Access control through central server (wat?)**

! re:claimID does **not** require a blockchain, is fully decentralized and allows asynchronous data access.
In a nutshell

\[ \text{re:claimID} = \text{Decentralized directory service} + \text{Cryptographic access control} \]
• Decentralized directory service

  • Secure name system with open name registration.
  • Idea “borrowed” from NameID.
  • Example: nslookup email.bob.org ⇒ “bob@example.com”
  • Our implementation uses the GNU Name System (GNS)
In a nutshell

- Decentralized directory service
  - Secure **name system** with open name registration.
  - Idea “borrowed” from NameID.
  - Example: nslookup email.bob.org ⇒ “bob@example.com”
  - Our implementation uses the **GNU Name System (GNS)**

- Cryptographic access control layer
  - Provided by GNS through encrypted and signed resource records.
  - Protects identity data from unwanted disclosure and allows users to enforce access control.
How does it work
Managing and publishing identity information

Decentralised directory

Publish encrypted identity data

User

Website

re:claimID
In GNS, a namespace is defined by a public/private EC key pair:

- $x$: Private key
- $P$: Public key
- $G$: Generator of the curve
- $n$: Group order

Records are encrypted and signed using keys derived from $(x, P)$. Encrypted records are published in a distributed hash table (under key $q$). Any peer is able to verify the signature as the corresponding derived public key is also published. Records can only be resolved and decrypted if the true identity and the label is known. ⇒ Namespaces cannot be enumerated and queries/responses cannot be observed. *Unless label and identity are known.*
The GNU Name System

- In GNS, a namespace is defined by a public/private EC key pair:
  - $x$: Private key
  - $P$: Public key
  - $G$: Generator of the curve
  - $n$: Group order
- Records are encrypted and signed using keys derived from $(x, P)$. 
The GNU Name System

- In GNS, a namespace is defined by a public/private EC key pair:
  - $x$: Private key
  - $P$: Public key
  - $G$: Generator of the curve
  - $n$: Group order

- Records are encrypted and signed using keys derived from $(x, P)$.
- Encrypted records are published in a distributed hash table (under key $q$).

⇒ Namespaces cannot be enumerated and queries/responses cannot be observed.

*Unless label and identity are known.*
In GNS, a namespace is defined by a public/private EC key pair:

- \( x \): Private key
- \( P \): Public key
- \( G \): Generator of the curve
- \( n \): Group order

Records are encrypted and signed using keys derived from \((x, P)\).

Encrypted records are published in a distributed hash table (under key \(q\)).

Any peer is able to verify the signature as the corresponding derived public key is also published.
In GNS, a namespace is defined by a public/private EC key pair:
- $x$: Private key
- $P$: Public key
- $G$: Generator of the curve
- $n$: Group order

Records are encrypted and signed using keys derived from $(x, P)$.

Encrypted records are published in a distributed hash table (under key $q$).

Any peer is able to verify the signature as the corresponding derived public key is also published.

Records can only be resolved and decrypted if the true identity and the label is known.
The GNU Name System

- In GNS, a namespace is defined by a public/private EC key pair:
  - $x$: Private key
  - $P$: Public key
  - $G$: Generator of the curve
  - $n$: Group order

- Records are encrypted and signed using keys derived from $(x, P)$. 
- Encrypted records are published in a distributed hash table (under key $q$).
- Any peer is able to verify the signature as the corresponding derived public key is also published.
- Records can only be resolved and decrypted if the true identity and the label is known.

$\Rightarrow$ Namespaces cannot be enumerated and queries/responses cannot* be observed.

*Unless label and identity are known.
Users may create a namespace \((x, P)\) and use it as a digital identity containing personal information:

<table>
<thead>
<tr>
<th>Label</th>
<th>Record Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l_{email})</td>
<td>ATTR</td>
<td>“email=<a href="mailto:alice@example.com">alice@example.com</a>”</td>
</tr>
<tr>
<td>(l_{name})</td>
<td>ATTR</td>
<td>“name=Alice Doe”</td>
</tr>
<tr>
<td>(l_{dob})</td>
<td>ATTR</td>
<td>“dob=1.3.1987”</td>
</tr>
</tbody>
</table>

where the labels are random secret values with high entropy.
Given a namespace $(x, P)$, we can treat labels as shared secrets in order to selectively disclose information.

\[ h := \text{Hash}(l_{\text{attr}}, P) \]
Given a namespace \((x, P)\), we can treat labels as shared secrets in order to selectively disclose information.

\[
h := \text{Hash}(l_{\text{attr}}, P)
\]

\[
\text{DHT key } \begin{cases} 
q := H(hP)
\end{cases}
\]
Given a namespace \((x, P)\), we can treat labels as shared secrets in order to selectively disclose information.

\[
h := \text{Hash}(l_{\text{attr}}, P)
\]

- **DHT key** \(q := H(hP)\)
- **Encryption** \(k := \text{HKDF}(l_{\text{attr}}, P)\)
  \(\text{Record} := \text{Enc}_k(\text{Data})\)

\[
d := h \cdot x \mod n
\]

\[
\text{Signature} = \text{Sig}_d(\text{Record})
\]
Publishing information

Given a namespace \((x, P)\), we can treat labels as shared secrets in order to selectively disclose information.

\[
h := \text{Hash}(l_{\text{attr}}, P)
\]

**DHT key** \[
q := H(hP)
\]

**Encryption** \[
k := \text{HKDF}(l_{\text{attr}}, P)
\]
\[
\text{Record} := \text{Enc}_k(\text{Data})
\]

**Signature** \[
d := h \cdot x \ mod \ n
\]
\[
\text{Signature} = \text{Sig}_d(\text{Record})
\]
Authorizing access

Decentralised directory

Grant access

User

Request authorisation

Website
For each authorized party, the user publishes reference records under the secret label.

• ticket can be shared with a third party in order to authorize access to email and dob.

• Indirection enables us to revoke tickets.

<table>
<thead>
<tr>
<th>Label</th>
<th>Record Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{email}$</td>
<td>ATTR</td>
<td>(\text{“email=<a href="mailto:alice@doe.com">alice@doe.com</a>”})</td>
</tr>
<tr>
<td>$l_{name}$</td>
<td>ATTR</td>
<td>(\text{“name=Alice Doe”})</td>
</tr>
<tr>
<td>$l_{dob}$</td>
<td>ATTR</td>
<td>(\text{“dob=1.3.1987”})</td>
</tr>
</tbody>
</table>
### Authorizing access

<table>
<thead>
<tr>
<th>Label</th>
<th>Record Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{email}$</td>
<td>ATTR</td>
<td>“email=<a href="mailto:alice@doe.com">alice@doe.com</a>”</td>
</tr>
<tr>
<td>$I_{name}$</td>
<td>ATTR</td>
<td>“name=Alice Doe”</td>
</tr>
<tr>
<td>$I_{dob}$</td>
<td>ATTR</td>
<td>“dob=1.3.1987”</td>
</tr>
<tr>
<td>$I_{ticket}$</td>
<td>ATTR_REF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATTR_REF</td>
<td>$I_{email}$</td>
</tr>
<tr>
<td></td>
<td>ATTR_REF</td>
<td>$I_{dob}$</td>
</tr>
</tbody>
</table>

- For each authorized party, the user publishes reference records under the secret label $I_{ticket}$.
- $I_{ticket}$ can be shared with a third party in order to authorize access to email and dob.
- Indirection enables us to revoke tickets.
Retrieve and decrypt attributes
Given an identity with public key $P$, we can retrieve references using $l_{\text{ticket}}$ and subsequently identity info from GNS.

$$h := \text{Hash}(l_{\text{ticket}}, P)$$
Given an identity with public key $P$, we can retrieve references using $l_{ticket}$ and subsequently identity info from GNS.

\[ h := \text{Hash}(l_{ticket}, P) \]

DHT key \[
\begin{cases}
q := H(hP)
\end{cases}
\]
Given an identity with public key $P$, we can retrieve references using $I_{\text{ticket}}$ and subsequently identity info from GNS.

$$h := \text{Hash}(I_{\text{ticket}}, P)$$

DHT key

$$q := H(hP)$$

Record decryption

$$k := \text{HKDF}(I_{\text{ticket}}, P)$$
$$\text{Data} := \text{Dec}_k(\text{Record})$$
Integration

- re:claimID implements the OpenID Connect protocol.
- For websites, it is just like integrating any other IdP (e.g. Google)
- For users, the authorization flow looks just like with any other OpenID Connect IdP.
Demo
Who sais that, anyway?
• Sometimes we need third party assurances to establish trust in identities.
Attestations

- Sometimes we need third party assurances to establish trust in identities.
- Currently, IdPs such as Facebook/Google implicitly provide this assurance (i.e. vouch for the truthfulness and correctness).
Attestations

• Sometimes we need third party assurances to establish trust in identities.
• Currently, IdPs such as Facebook/Google implicitly provide this assurance (i.e. vouch for the truthfulness and correctness).
• Claim: Those parties are not actually the authorities over (most of) your personal data! Examples:
  • Real name (State/Self-asserted/Other organization)
  • Phone number (Provider)
  • Address (State/Self-asserted)
  • Citizenship (State)
  • Age (State)
  • Email address (Mail provider)
• What users actually need is a **collection of credentials**.
Attestations

- What users actually need is a collection of credentials.
- Those credentials are issued by a variety of different entities, including the user.
• What users actually need is a **collection of credentials**.
• Those credentials are issued by a **variety of different entities**, including the user.
• Credentials are ideally **preserving the privacy** of the individual, e.g. using zero-knowledge proofs.
What users actually need is a **collection of credentials**.

Those credentials are issued by a **variety of different entities**, including the user.

Credentials are ideally **preserving the privacy** of the individual, e.g. using zero-knowledge proofs.

Those ideas are already finding their way into standards:

- W3C: “Verifiable Credentials”
- OpenID Connect: “Aggregated Claims”
• What users actually need is a collection of credentials.
• Those credentials are issued by a variety of different entities, including the user.
• Credentials are ideally preserving the privacy of the individual, e.g. using zero-knowledge proofs.
• Those ideas are already finding their way into standards:
  • W3C: “Verifiable Credentials”
  • OpenID Connect: “Aggregated Claims” ← working on it.
Using re:claimID
Installing re:claimID

1. Install the webextension:
   https://addons.mozilla.org/firefox/addon/reclaimid/
Installing re:claimID

1. Install the webextension:
   https://addons.mozilla.org/firefox/addon/reclaimid/
2. Install GNUnet $\geq 0.11.6$
Installing re:claimID

Get help installing GNUnet and/or re:claimID at our workshop today!

- Right after this.
- Time: 12:15 PM – 15:00 PM
- Location: Seminarraum
Summary
Status

- Demo websites exist:
  - https://demo.reclaim-identity.io
  - https://eusec.clouditor.io
- Roadmap:
  - User-friendly packaging (of GNUnet)
  - Ship GNUnet inside browser plugin (yes, that might even work).
  - “1.0” by end of 2019
Questions?

https://reclaim-identity.io
https://gnunet.org

schanzen@aisec.fraunhofer.de
6665 201E A925 7CC6 8FDE 77E8 8433 5131 EA3D ABF0
– or –
schanzen@gnunet.org
3D11 063C 10F9 8D14 BD24 D147 0B09 98EF 86F5 9B6A

2. Martin Schanzenbach, Georg Bramm, Julian Schütte. reclaimID: Secure, Self-Sovereign Identities Using Name Systems and Attribute-Based Encryption. 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom), 2018