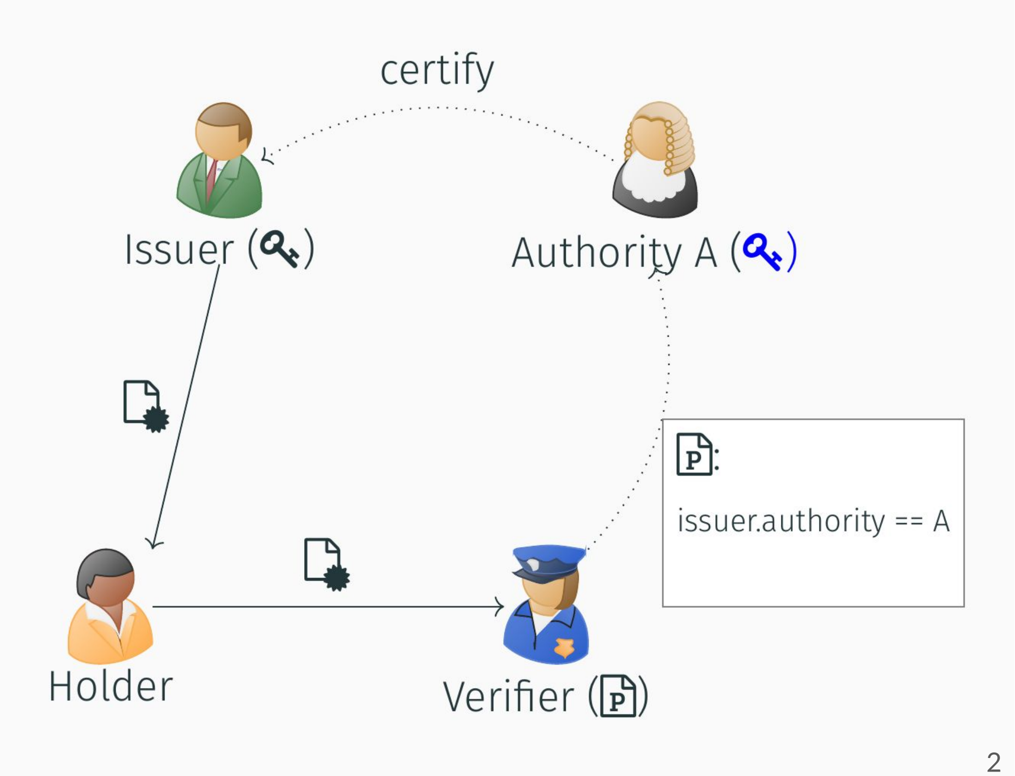
Adapting the *TPL* Trust Policy Language for a Self-Sovereign Identity World

Lukas Alber, **Stefan More** Graz University of Technology
Sebastian Mödersheim Technical University of Denmark
Anders Schlichtkrull Aalborg University Copenhagen

Context: **Trust Policies**



The *TPL* Trust Policy Language [MSW$^+$19]

Originally created for the H2020 project LIGHTest

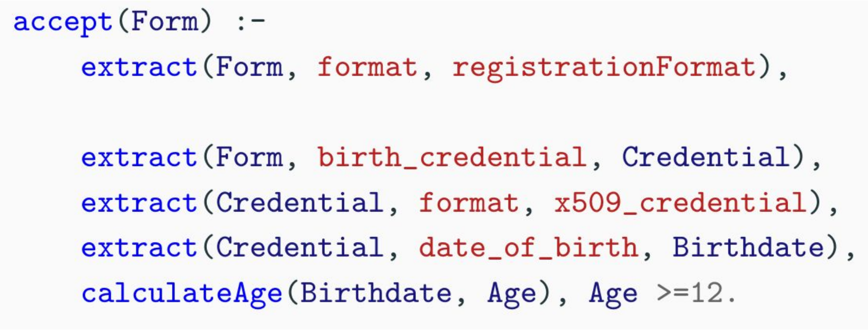
Design Goals:

- Support of various Trust Management concepts
- Modularity
- Declarativity and expressive power
- Formal precision and accountability

The *TPL* Trust Policy Language

- Based on horn clauses (Classical Logic, **Prolog-like syntax**)
- **Built-in predicates** for trust management concepts
- **Formats** provide extensible parser system
  (custom formats based on, e.g., XMLDsig/XAdES, PAdES, CAdES, ...)

- Isabelle-verified Verifier RPx

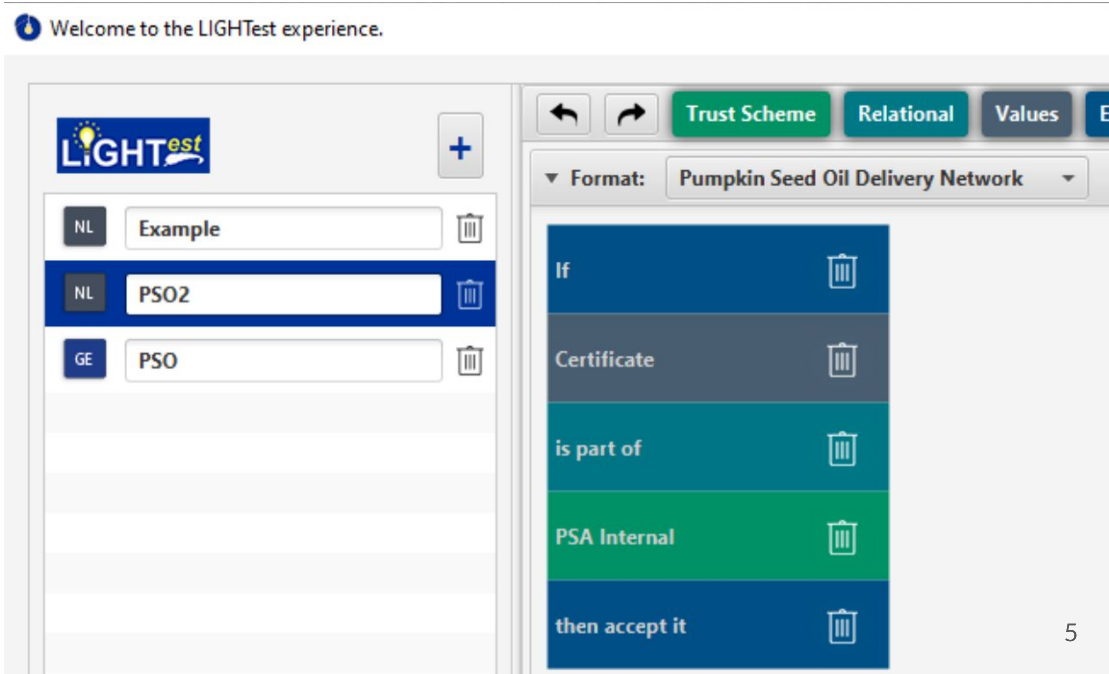


```
accept(Form) :-
    extract(Form, format, registrationFormat),

    extract(Form, birth_credential, Credential),
    extract(Credential, format, x509_credential),
    extract(Credential, date_of_birth, Birthdate),
    calculateAge(Birthdate, Age), Age >=12.
```

Natural-language and graphical *TPL* editors [MN19, WP19]

Enables authoring of policies
by (non-technical) domain experts.

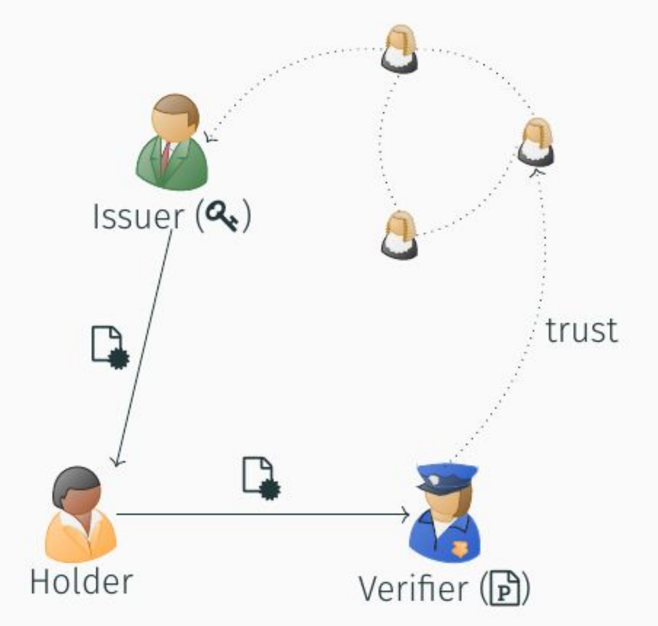


Welcome to the LIGHTest experience.

Background: **Distributed Ledgers (DLs)**

“Blockchain”

- Data store on distributed nodes
- Nodes use consensus protocol [Nak08, XZLH20]
- Trust distributed among nodes



Background: **Self-Sovereign Identity (SSI)**

- **Identity management model** [ZZS14]
- Users in sovereign/full control of their data

- New (proposed) standards:
  - **Verifiable Credentials** (W3C VCs) [SLC19]
  - **Decentralized Identifiers** (W3C DIDs) [RSL+ 21]
    - DID Documents, DID Registries, ...

Vision: *SSI TPL*

SSI TPL: **Motivation**

- We introduced *TPL* for a pre-SSI world
- No support for SSI concepts and distributed trust registries

- → Extend TPL with **support for DL, SSI** (DIDs and VCs)
- Enable new concepts to increase privacy

Building block for automating global trust management.

SSI TPL: **Goals**

- Adapt *TPL* for the SSI world without modifying the syntax
- Enable use of SSI concepts and trust sources in TPL

- Support both LIGHTest concepts (eIDAS, ...)
  and SSI concepts in the same policy

Example Use Case

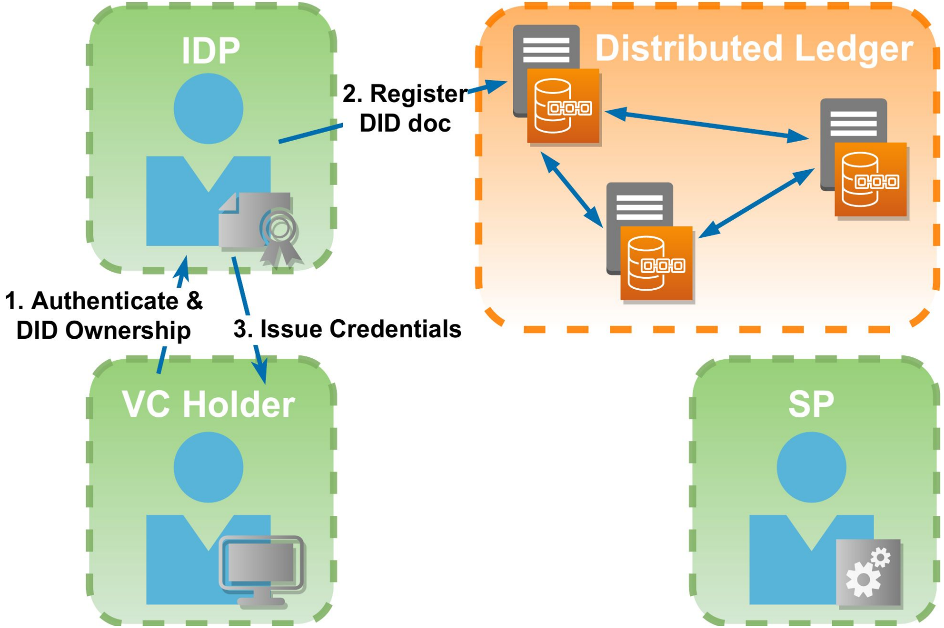
A pan-european portal
where teenagers can have supervised discussions
about the future of the continent.

In our example
a teenager registers at the portal.

Roles:

VC Holder = Verifiable Credential Holder = The Teenager

Flow



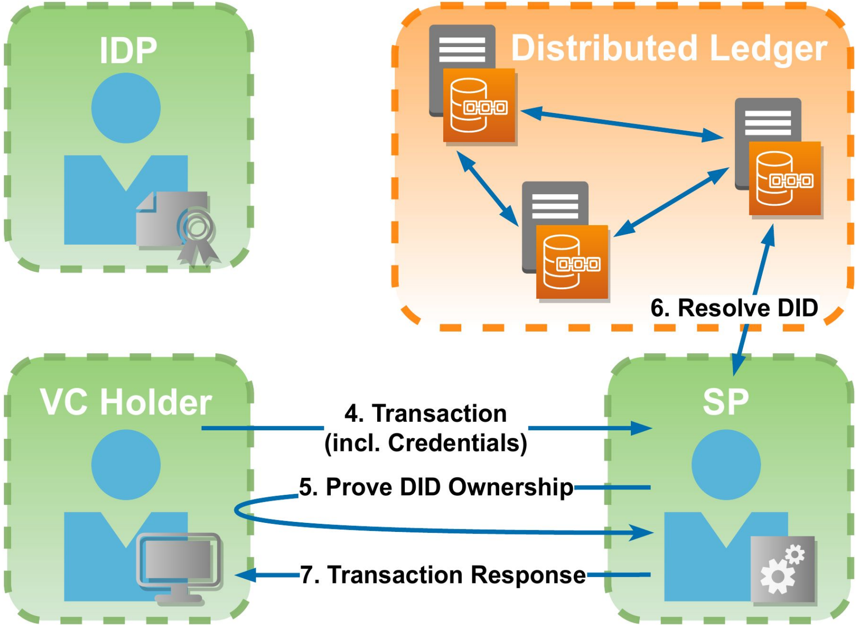
Flow

Roles:

VC Holder = Verifiable Credential Holder = The Teenager

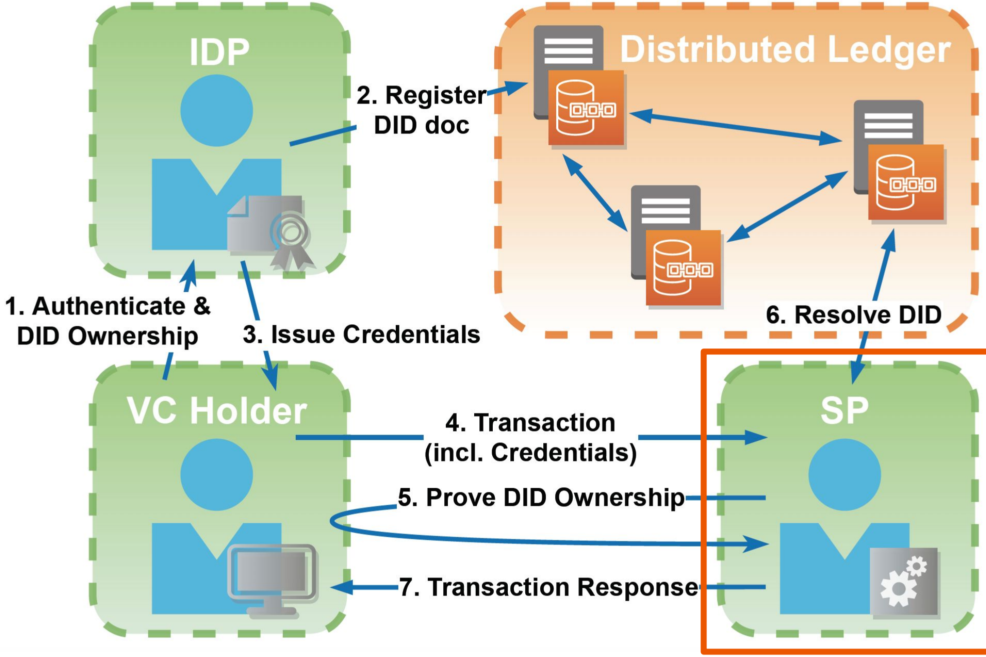
SP = Service Provider = The discussion portal

Transaction = The registration of the teenager at the portal



Flow

Roles:
VC Holder = Verifiable Credential Holder = The Teenager
SP = Service Provider = The discussion portal
Transaction = The registration of the teenager at the portal



SP runs the
TPL Interpreter.

Concept idea

Enable support for SSI in TPL by introducing

- **new formats**, and
- a **new built-in predicate.**

Let us see how!

New formats

`ssi_credential`: format for Verifiable Credential (VC).

We can check if some given data follows this format:

```
    extract(Credential, format, ssi_credential),
```

Then we can extract attributes:

```
    extract(Credential, date_of_birth, Birthdate),
```

New formats

`ssi_credential`: format for Verifiable Credential (VC).

We can check if some given data follows this format:

```
    extract(Credential, format, ssi_credential),
```

Then we can extract attributes:

```
    extract(Credential, date_of_birth, Birthdate),
```

`ssi_diddoc:` format for DID Documents.

We can check if some given data follows this format:

```
    extract(DIDDoc, format, ssi_diddoc),
```

Then we can extract attributes:

```
    extract(DIDDoc, pk, PK),
```

New formats

`ssi_credential`: format for Verifiable Credential (VC).

We can check if some given data follows this format:

```
    extract(Credential, format, ssi_credential),
```

Then we can extract attributes:

```
    extract(Credential, date_of_birth, Birthdate),
```

`ssi_diddoc:` format for DID Documents.

We can check if some given data follows this format:

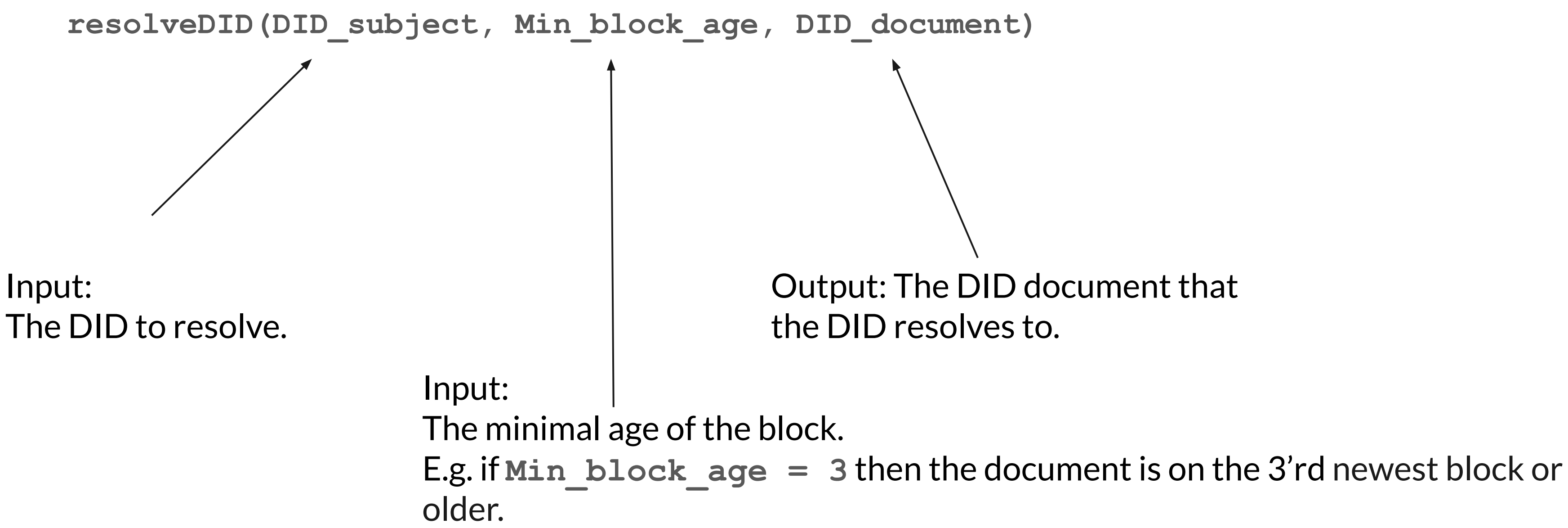
```
    extract(DIDDoc, format, ssi_diddoc),
```

Then we can extract attributes:

```
    extract(DIDDoc, pk, PK),
```

Note: The precise set of attributes depends on the context.

New built-in predicate



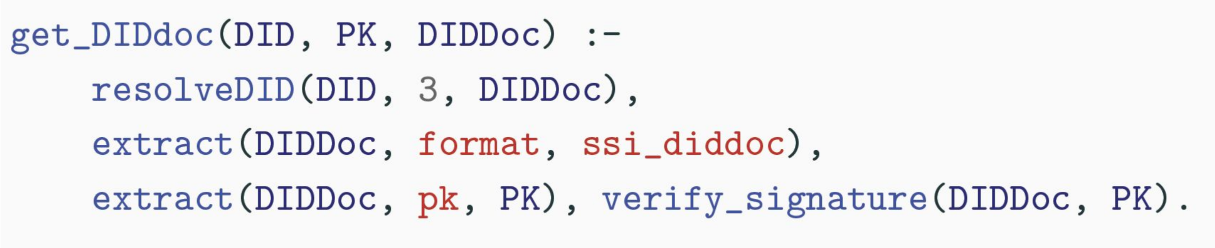
```
resolveDID(DID_subject, Min_block_age, DID_document)
```

Input:
The DID to resolve.

Input:
The minimal age of the block.
E.g. if `Min_block_age = 3` then the document is on the 3'rd newest block or older.

Output: The DID document that the DID resolves to.

Building a predicate for the SSI world

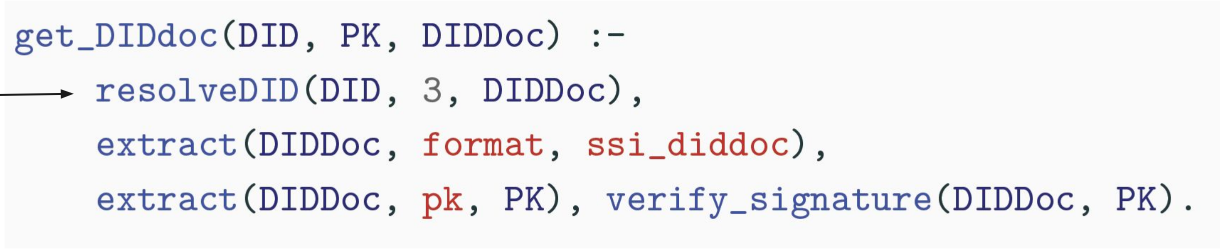


```
get_DIDdoc(DID, PK, DIDDoc) :-
    resolveDID(DID, 3, DIDDoc),
    extract(DIDDoc, format, ssi_diddoc),
    extract(DIDDoc, pk, PK), verify_signature(DIDDoc, PK).
```

Resolves a DID to get its public key and DID document.
And checks that the DID document is correctly signed.

Building a predicate for the SSI world

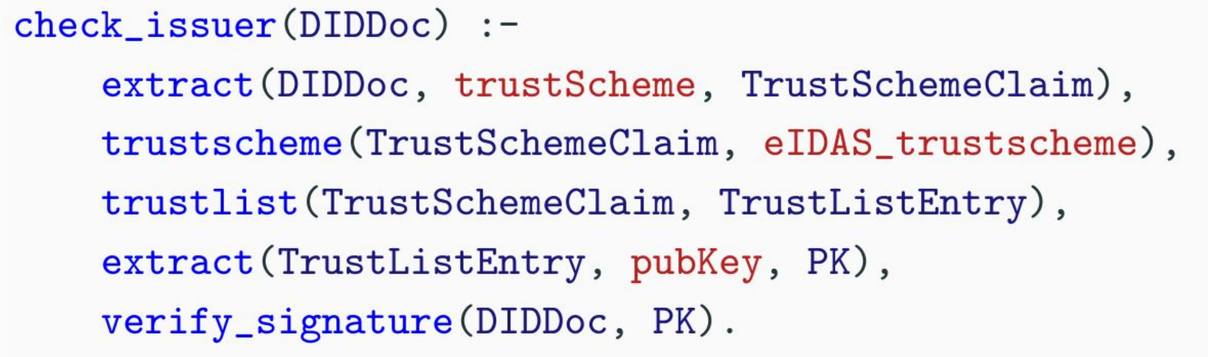
From the SSI world



```
get_DIDdoc(DID, PK, DIDDoc) :-
    resolveDID(DID, 3, DIDDoc),
    extract(DIDDoc, format, ssi_diddoc),
    extract(DIDDoc, pk, PK), verify_signature(DIDDoc, PK).
```

Resolves a DID to get its public key and DID document.
And checks that the DID document is correctly signed.

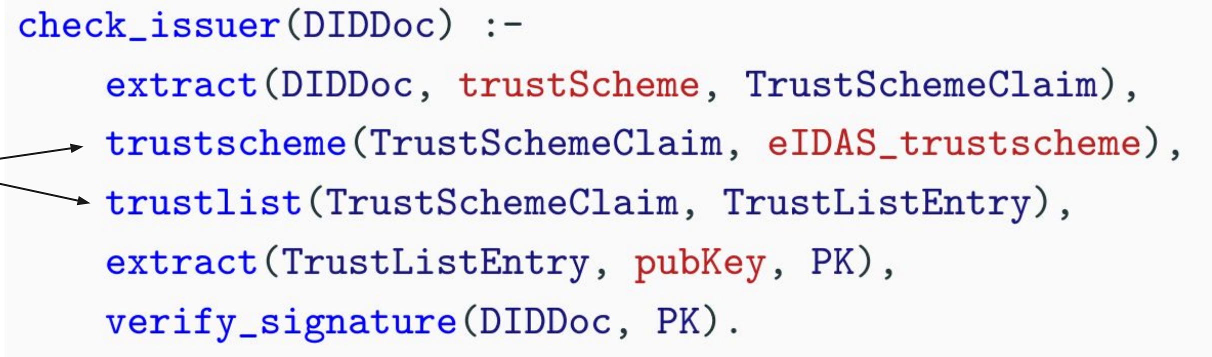
Building a predicate for the eIDAS world



```
check_issuer(DIDDoc) :-
    extract(DIDDoc, trustScheme, TrustSchemeClaim),
    trustscheme(TrustSchemeClaim, eIDAS_trustscheme),
    trustlist(TrustSchemeClaim, TrustListEntry),
    extract(TrustListEntry, pubKey, PK),
    verify_signature(DIDDoc, PK).
```

Checks that a credential’s Issuer is eIDAS qualified
using the trust scheme claim in their DID document.

Building a predicate for the eIDAS world

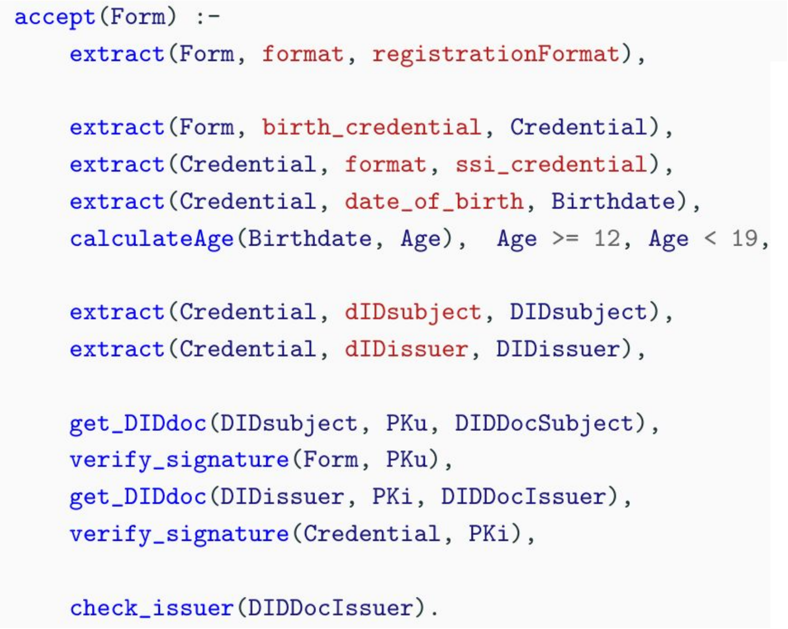


```
check_issuer(DIDDoc) :-
    extract(DIDDoc, trustScheme, TrustSchemeClaim),
    trustscheme(TrustSchemeClaim, eIDAS_trustscheme),
    trustlist(TrustSchemeClaim, TrustListEntry),
    extract(TrustListEntry, pubKey, PK),
    verify_signature(DIDDoc, PK).
```

From the eIDAS world

Checks that a credential’s Issuer is eIDAS qualified
using the trust scheme claim in their DID document.

A policy using both predicates



```
accept(Form) :-
    extract(Form, format, registrationFormat),

    extract(Form, birth_credential, Credential),
    extract(Credential, format, ssi_credential),
    extract(Credential, date_of_birth, Birthdate),
    calculateAge(Birthdate, Age),  Age >= 12, Age < 19,

    extract(Credential, dIDsubject, DIDsubject),
    extract(Credential, dIDissuer, DIDissuer),

    get_DIDdoc(DIDsubject, PKu, DIDDocSubject),
    verify_signature(Form, PKu),
    get_DIDdoc(DIDissuer, PKi, DIDDocIssuer),
    verify_signature(Credential, PKi),

    check_issuer(DIDDocIssuer).
```

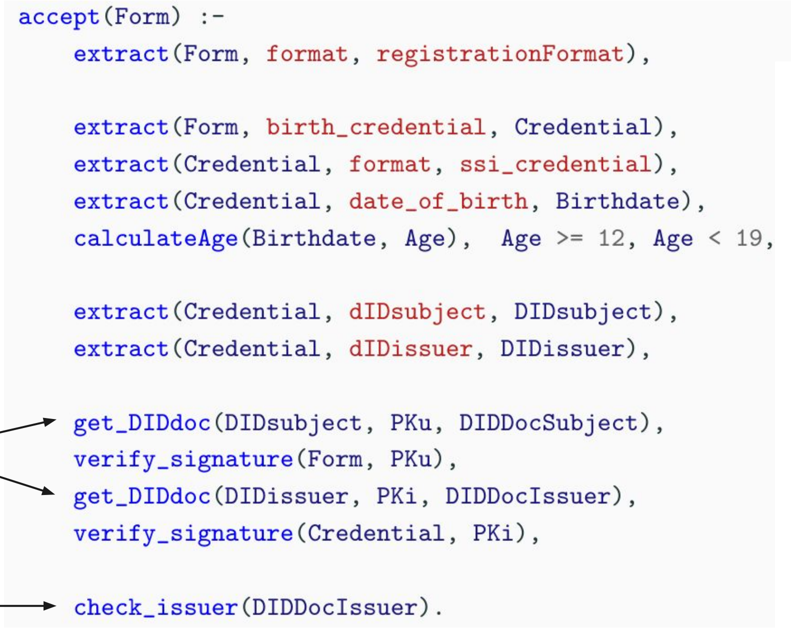
The credential claims that the subject is a teenager.

Extract DID of subject and issuer.

Get and check the corresponding DID Documents, and verify signatures.

Check that the issuer is eIDAS qualified

A policy using both predicates



```
accept(Form) :-
    extract(Form, format, registrationFormat),

    extract(Form, birth_credential, Credential),
    extract(Credential, format, ssi_credential),
    extract(Credential, date_of_birth, Birthdate),
    calculateAge(Birthdate, Age),  Age >= 12, Age < 19,

    extract(Credential, dIDsubject, DIDsubject),
    extract(Credential, dIDissuer, DIDissuer),

    get_DIDdoc(DIDsubject, PKu, DIDDocSubject),
    verify_signature(Form, PKu),
    get_DIDdoc(DIDissuer, PKi, DIDDocIssuer),
    verify_signature(Credential, PKi),

    check_issuer(DIDDocIssuer).
```

The credential claims that the subject is a teenager.

Extract DID of subject and issuer.

Get and check the corresponding DID Documents, and verify signatures.

Check that the issuer is eIDAS qualified

From the SSI world

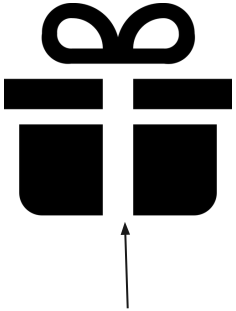
From the eIDAS world

Accountability and verification

We want to make trust decisions accountable:

- A business transaction can enter a legal dispute.
- Then we need to look at what happened.

Idea: A small “package”



Policy
+
Trust information from a server (e.g. identity documents)

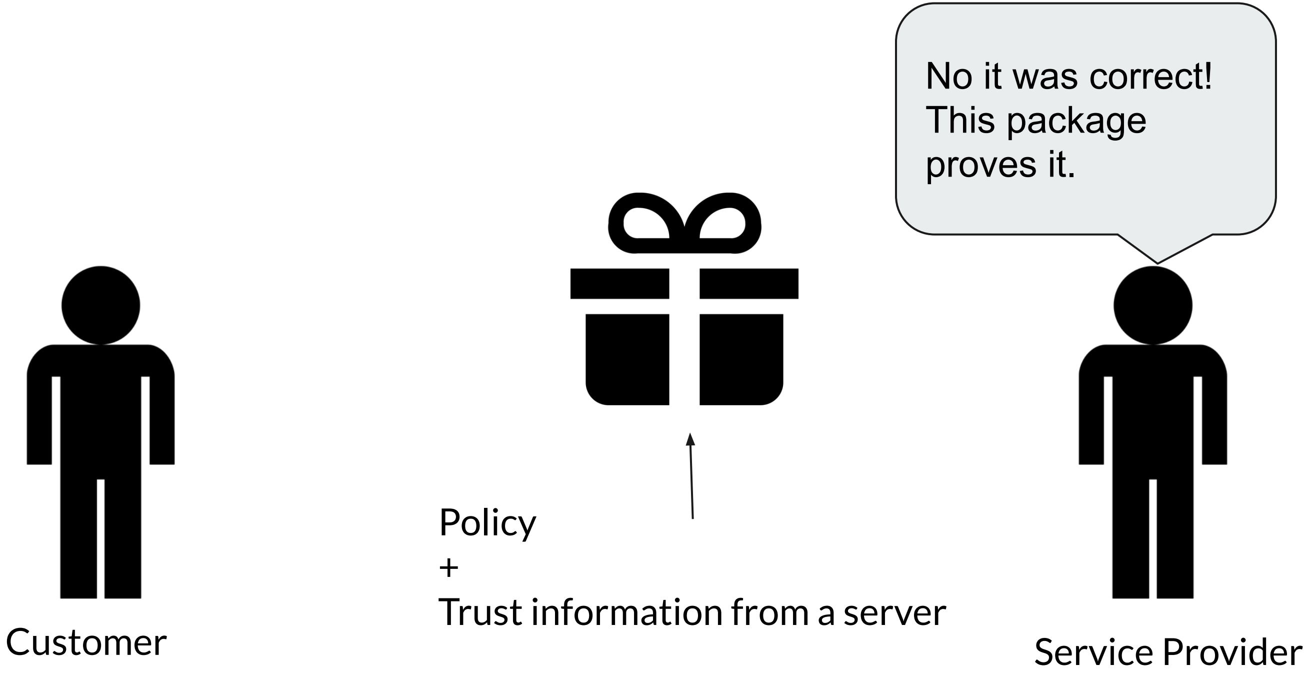
Accountability and verification

Your trust decision was wrong!

Customer

Service Provider

Accountability and verification



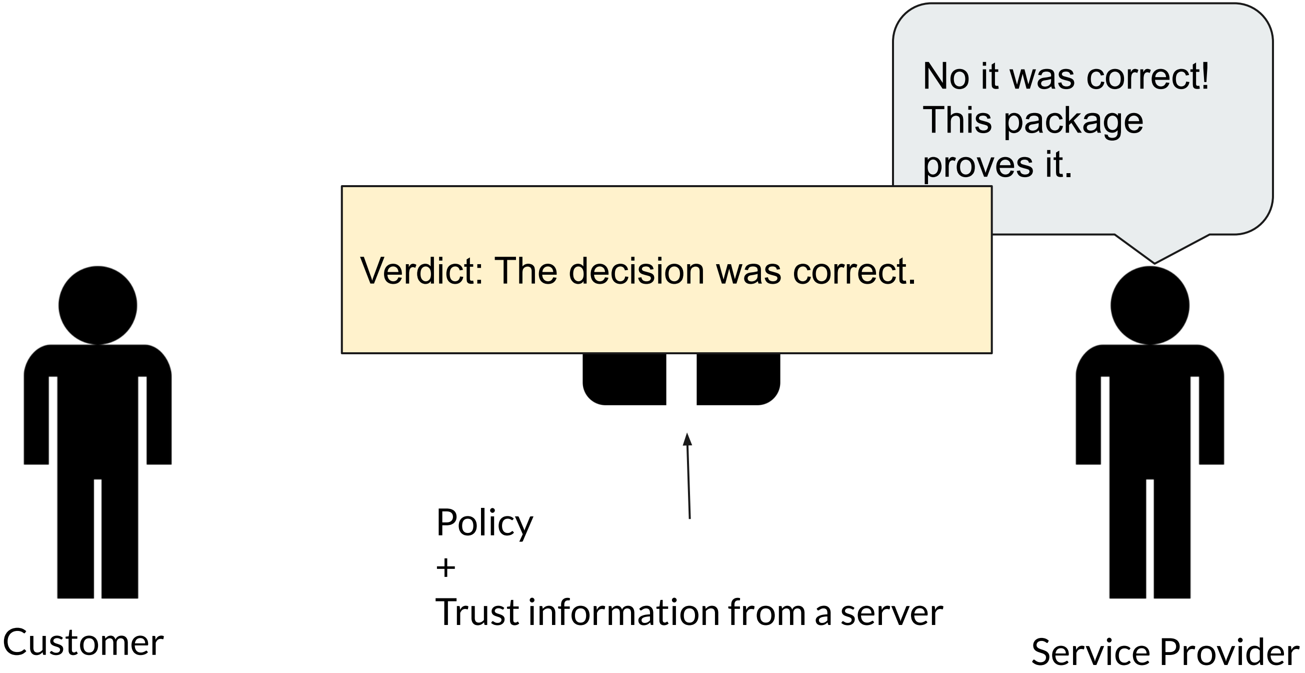
No it was correct! This package proves it.

Customer

Policy
+
Trust information from a server

Service Provider

Accountability and verification



No it was correct! This package proves it.

Verdict: The decision was correct.

Policy
+
Trust information from a server

Accountability and verification

Problem: There is room for a counter argument!

Let us take a look again.

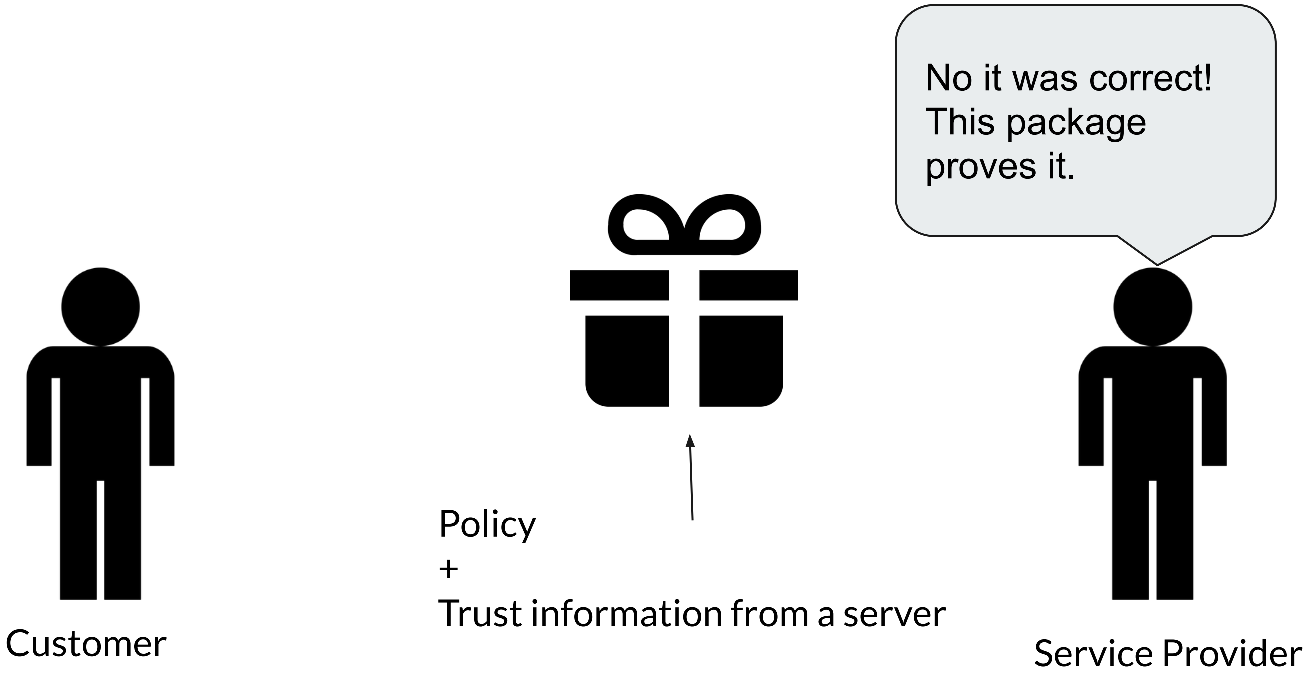
Accountability and verification

Your trust decision was wrong!

Customer

Service Provider

Accountability and verification



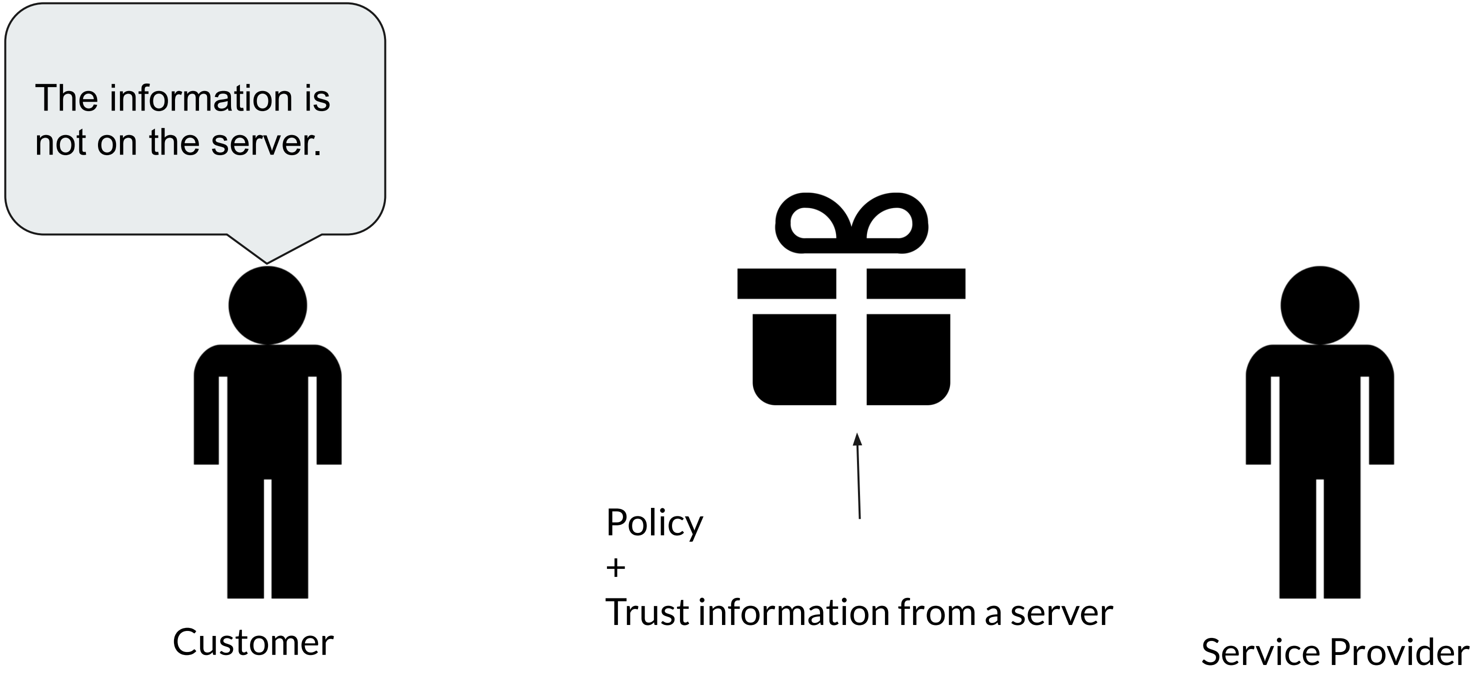
No it was correct! This package proves it.

Policy
+
Trust information from a server

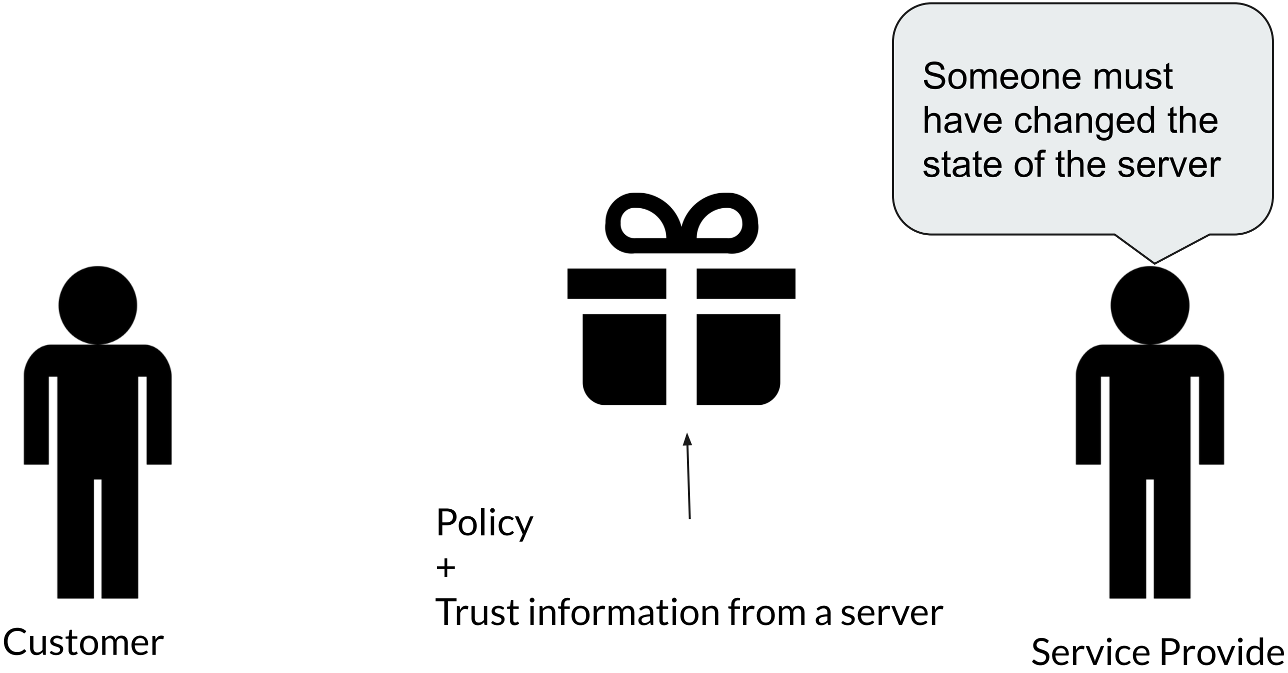
Customer

Service Provider

Accountability and verification


The information is not on the server.
Policy
+
Trust information from a server
Customer
Service Provider

Accountability and verification



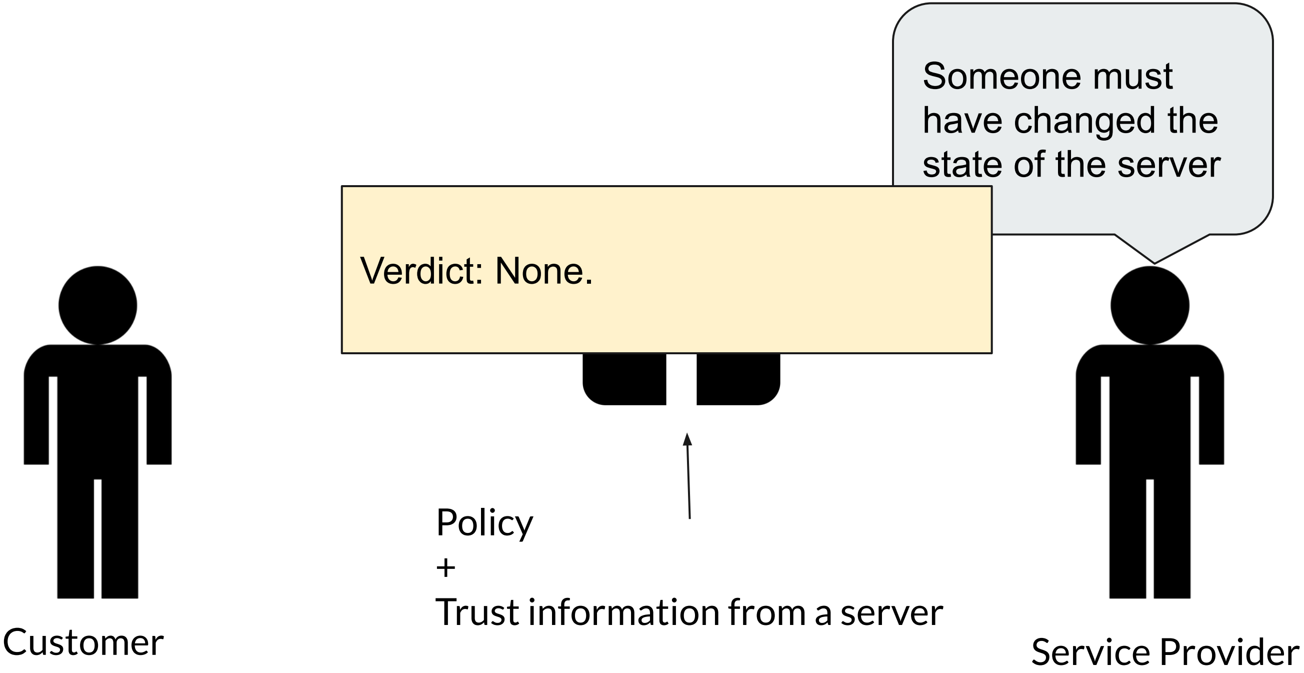
Someone must have changed the state of the server

Customer

Policy
+
Trust information from a server

Service Provider

Accountability and verification


Someone must have changed the state of the server
Verdict: None.
Policy
+
Trust information from a server
Customer
Service Provider

Accountability and verification

Suggested solution:

By having trust information on a ledger we can see the history.

Since the ledger is distributed we do not have to blindly trust some single organization’s records.

But the information should be on an “old enough” block!

```
resolveDID(DID_subject, Min_block_age, DID_document)
```

Should be “sufficiently large”!

Accountability and verification

There is also another challenge:

- The TPL interpreter makes decisions.
- The TPL interpreter is software.
- Software can have bugs.
- The TPL interpreter could in principle -- despite all of our testing -- have a bug.

Accountability and verification

What if a transaction was accepted only because of a bug in the TPL interpreter's reasoning?

The TPL interpreter can make a proof certificate: (p, q, b)

- p: TPL policy
- q: query
- b: record of calls to the built-in predicates and results

We can then check the reasoning using RPx:

- RPx is a first-order theorem prover
- RPx is implemented independently from the TPL interpreter
- RPx's inference engine is verified in the Isabelle/HOL proof assistant
    - Thus, bugs in RPx are unlikely.

Accountability and verification

Will the proof certificates and RPx work for SSI?

Yes!

The proof certificate is independent of the concrete formats and predicates.
RPx is independent of the concrete concrete formats and predicates.

Therefore it works for SSI and DL “out of the box”.

Future work and outlook

Implementation

- The obvious next step!

Range proofs

- In our example, the user sends his birth date to the service provider.
- But only one bit of information is needed: “Is the user a teenager or not?”
- Range proofs can solve this.

Conclusion

We have shown how the SSI concepts fit in TPL:

- Requires two new formats and a new built-in predicate.
- These can be used as components in other predicates.

There are several advantages:

- This allows for policies that rely on identities from the SSI world.
- Trust information is stored on a ledger.
- RPx gives us an independent pair of eyes for the TPL interpreter.
- Policies can simultaneously rely on SSI and eIDAS.

The Paper

Alber, L., More, S., Mödersheim, S. & Schlichtkrull, A. (2021).

Adapting the *TPL* Trust Policy Language for a Self-Sovereign Identity World.

In: Roßnagel, H., Schunck, C. H. & Mödersheim, S. (Hrsg.),
Open Identity Summit 2021.
Bonn: Gesellschaft für Informatik e.V.. (S. 107-118).

https://dl.gi.de/handle/20.500.12116/36506

The other Paper

Mödersheim S., Schlichtkrull A., Wagner G., More S., Alber L. (2019).

TPL: A Trust Policy Language.

In: Meng W., Cofta P., Jensen C., Grandison T. (eds) Trust Management XIII.
IFIPTM 2019. IFIP Advances in Information and Communication Technology, vol 563. Springer, Cham.

https://doi.org/10.1007/978-3-030-33716-2_16

https://backend.orbit.dtu.dk/ws/files/198699175/IFIPTM.pdf

Thank you / Tak / Danke!

Anders Schlichtkrull | andsch@cs.aau.dk
Stefan More | smore@iaik.tugraz.at

This work was supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreements No. 871473 (**KRAKEN**), No. 959072 (**mGov4EU**), and No. 830929 (**CyberSec4Europe**), as well as the Sapere-Aude project “**Composec**: Secure Composition of Distributed Systems'”, grant 4184-00334B of the Danish Council for Independent Research.

References

[MN19] Sebastian Alexander Mödersheim and Bihang Ni. GTPL: A graphical trust policy language. In Open Identity Summit, volume P-293 of LNI, pages 107–118. GI, 2019.

[MSW+19] Sebastian Mödersheim, Anders Schlichtkrull, Georg Wagner, Stefan More, and Lukas Alber. TPL: A trust policy language. In IFIPTM, volume 563 of IFIP Advances in Information and Communication Technology, pages 209–223. Springer, 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008.

[RSL+ 21] Drummond Reed, Manu Sporny, Dave Longley, Christopher Allen, Ryan Grant, and Markus Sabadello. Decentralized Identifiers (DIDs) v1.0. W3C working draft, W3C, 2021.

[SLC19] Manu Sporny, Dave Longley, and David Chadwick. Verifiable Credentials Data Model 1.0. W3C recommendation, W3C, 2019.

[WP19] Stephanie Weinhardt and Doreen St. Pierre. Lessons learned - conducting a user experience evaluation of a trust policy authoring tool. In Open Identity Summit, volume P-293 of LNI, pages 185–190. GI, 2019.

[XZLH20] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks. IEEE Commun. Surv. Tutorials, 22(2):1432–1465, 2020.

[ZZS14] Bernd Zwattendorfer, Thomas Zefferer, and Klaus Stranacher. An overview of cloud identity management-models. In WEBIST (1), pages 82–92. SciTePress, 2014.

Sources

Person-clipart: https://publicdomainvectors.org/en/free-clipart/Man-icon-vector-image/9384.html

Gift box-clipart: https://publicdomainvectors.org/en/free-clipart/Gift-box-icon/59762.html