Enabling Decentralised Identifiers and Verifiable Credentials for Constrained Internet-of-Things Devices using OAuth-based Delegation

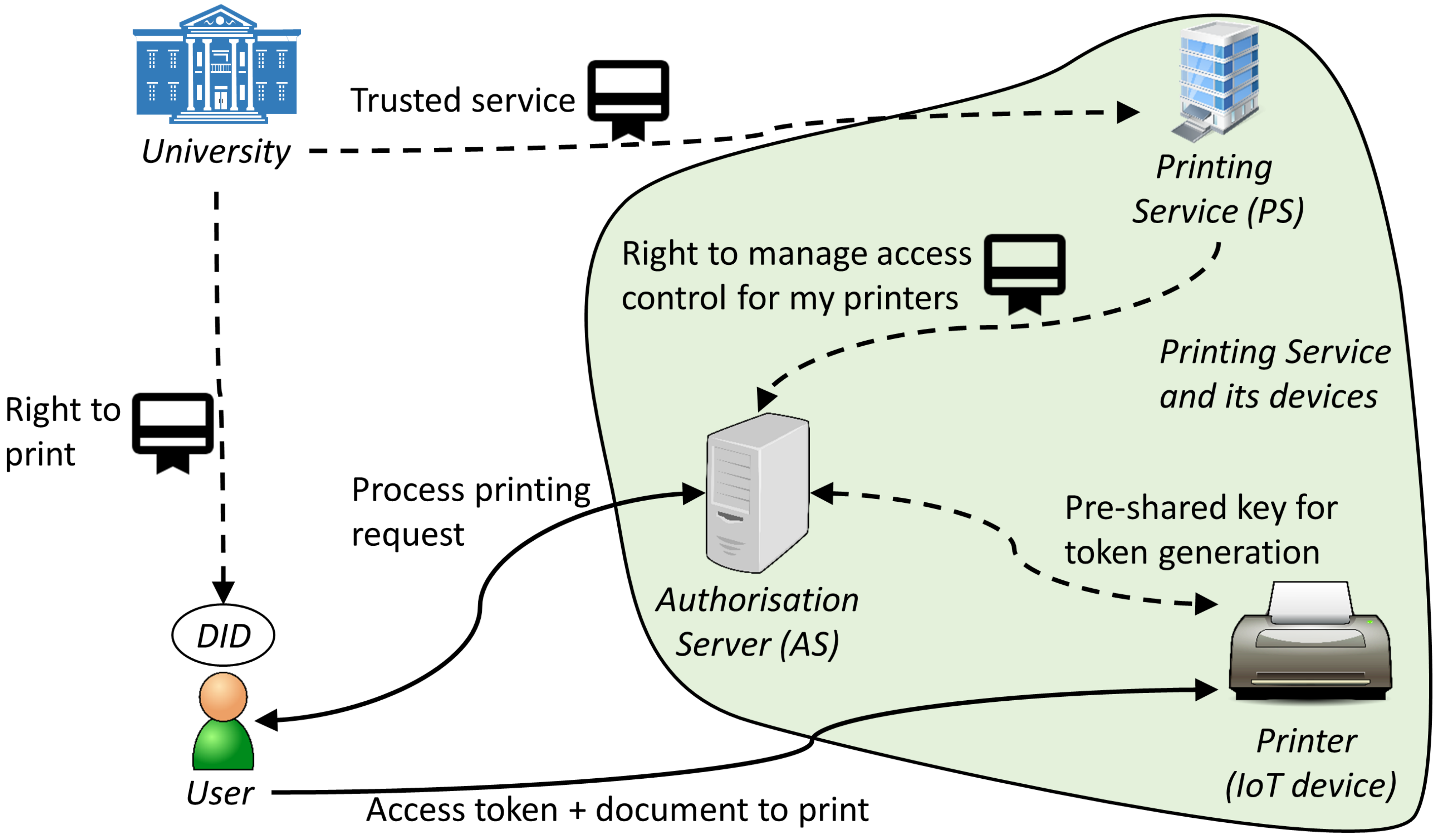
Dmitrij Lagutin, dmitrij.lagutin@aalto.fi
Aalto University, Finland
In co-operation with
Yki Kortesniemi, Nikos Fotiou, Vasilios A. Siris

Workshop on Decentralised IoT Systems
and Security (DISS)
San Diego, USA, 24.2.2019

Motivation: Identifiers and IoT

- It should be possible to use services and devices while preserving privacy and preventing tracking
- Current identifier and certificate solutions have several problems
  - Different identifier for each service, lack of interoperability
  - Social logins: lack of privacy and control by the user
  - Very complicated to provide privacy-preserving proofs online
- From privacy's point of view, digital identifiers should provide:
  - Self-sovereignty (owner controls the identifier)
  - Ability to change identifiers at will
  - Anonymity

Use Case: Printing at University



Requirements of Use Case

- User (visiting lecturer) wants to print a document before the lecture in a secure way
  - User does not have a university user account
  - Printers are managed by a third-party printing service, which collaborates with the university
- User should stay anonymous as much as possible
  - Printing Service (PS), Authorisation Server (AS), or Printer will never learn user's real identity or able to track user
- User and printing service need mutually authenticate each other:
  - Printing Service is trusted by University
  - Authorisation Server is trusted by Printing Service
  - User has right to print from University

Decentralised Identifiers and Verifiable Credentials

- Decentralised Identifiers (DIDs) aim to provide *self-sovereignty*
  - DIDs can be created by the user without dependence on any third party, hence a large number of DIDs can be used (even different one for each transaction)
  - Often derived from key pair, e.g.: *did:sov:3k9dg356wdcj5gf2k9bw8kfg7a*
- With *verifiable credentials* (VCs), owner of identifier can “prove” something (e.g. date of birth, degree) about themselves
  - Selective disclosure: disclose only part of the information present in credential
  - Zero-knowledge Proofs (ZKP) allow one to prove of, e.g., being over certain age without revealing real age
- IoT devices may not be able to use public-key cryptography (resource constrains, lack of entropy, cost of upgrading, etc.)
- **How to use privacy-enabling properties of DIDs and VCs with existing constrained IoT devices?**

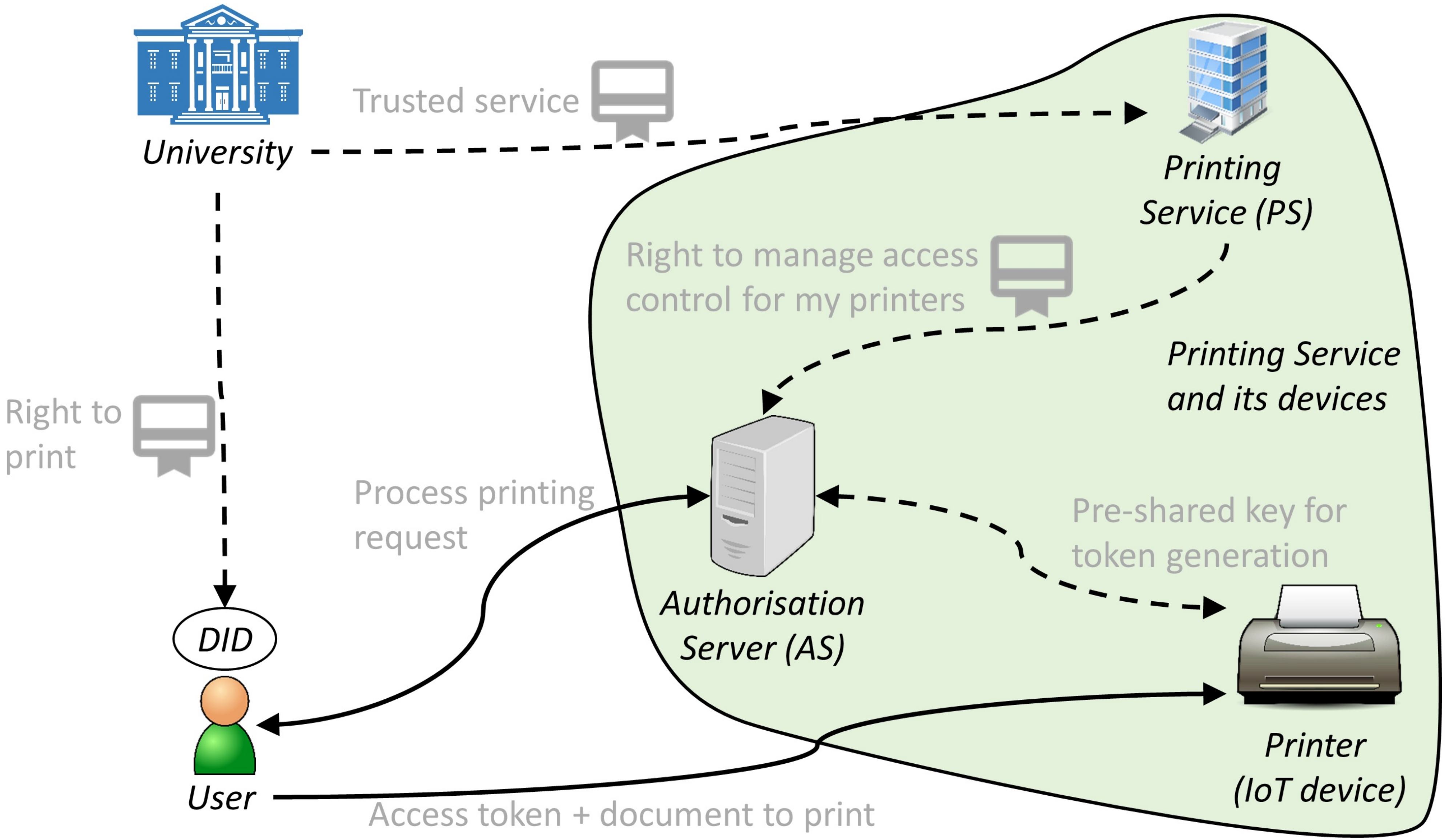
Existing Solutions for IoT Authentication and Authorisation

- OAuth 2.0 allows a client to obtain access to protected resource, residing on resource server (RS)
  - Access control is managed by authorisation server (AS), which issues access tokens
  - OAuth does not define used authentication solution
  - ACE extension for constrained IoT devices: allows usage of proof-of-possesion tokens that are based on pre-shared key
- Some technologies are not relevant for this use case
  - OpenID Connect: functionality is provided by DIDs and VCs
  - User Managed Access (UMA) 2.0: not always suitable for constrained devices

Delegating DID processing with OAuth

- Authorisation Server (AS) can act as a bridge between OAuth and DIDs
  - All actors except the device (printer) utilize DIDs and VCs for mutual authentication
  - Printer delegates DID processing to AS
- AS issues proof-of-possesion access tokens to client (Lecturer), after authentication has been performed (ACE-OAuth)
  - Lecturer uses the access token to access the printer

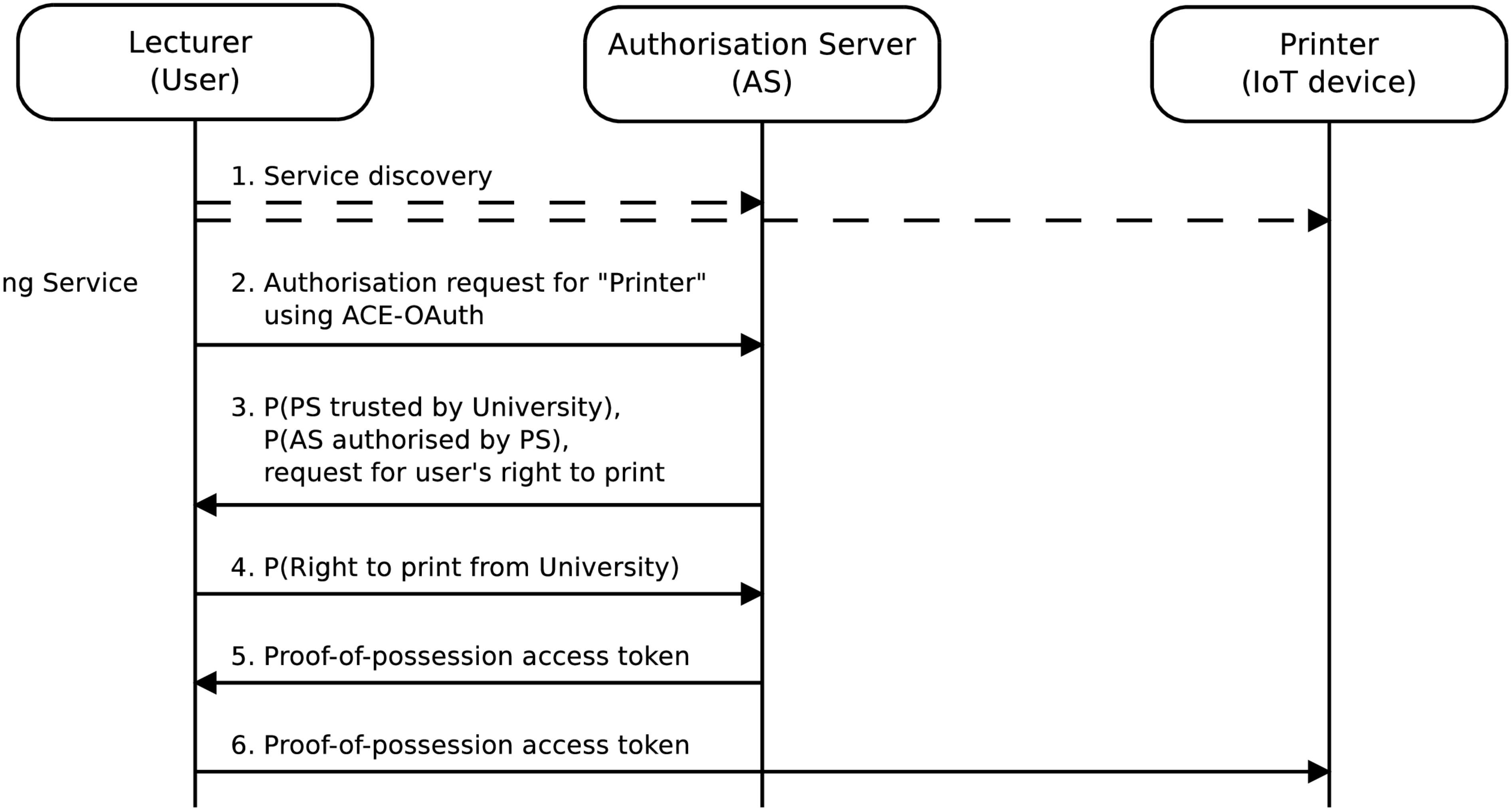
Printing at University: Actors



Message Flow

PS - Printing Service

P - Proof



Implementation

- The described solution has been implemented using Sovrin DID scheme (Hyperledger Indy) and OAuth2 server
  - User receives credentials from a Hyperledger Indy instance
  - User contacts OAuth server as usual
  - OAuth server generates a proof request containing a nonce
  - User generates a proof based on credentials using the nonce
  - Communication continues using standard OAuth protocol
- The source code will be made available before the publication of the paper

Comparison with Existing Solutions

- Using decentralised identifiers improves privacy

	X.509 Certificates	***DID + VC***
Granularity	Coarse	Fine-grained
Duration	Usually long	Short or long
Processing	By humans	Machine-readable

- Printing service or the printer will never learn real identify of user
  - User can change DIDs frequently to protect against correlations attacks
- Proposed solution is compatible with and complementary to OAuth and its extensions
  - Provides mutual authentication, decouples resource server from AS, can provide trusted AS discovery
  - No modification to the actual device (printer) necessary

Conclusions

- Decentralised identifiers and verifiable credentials improve privacy in several situations
  - Open standards, allowing easy deployment and adoption across organisations and industries
- Delegation allows constrained OAuth-capable devices to take advantage of DIDs and VCs
  - Without any modifications to existing devices
- We have implemented a proof-of-concept solution which will be released as open source

The research reported here has been partially undertaken in the context of projects SOFIE (Secure Open Federation for Internet Everywhere), which has received funding from European Unions Horizon 2020 research and innovation programme under grant agreement No. 779984, and TrustNet (Trust Network for Distributed Personal Data Management), which has received funding from Business Finland under grant No. 3387/31/2017.

Backup slide: Message Flow

- User and printing service mutually authenticate each other using proofs:
  - Printing Service is trusted by University
  - Authorisation Server is trusted by Printing Service
  - User has right to print from University
- Afterwards, proof-of-possesion access token, derived from pre-shared key, is issued using standard ACE-Oauth
- Message flow can be optimised by transmitting proofs during TLS handshake, utilising Encrypted Server Name Indication extension (TLS 1.3)

Verifiable Credential Example


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://example.com/examples/v1"
  ],
  "id": "http://example.gov/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu",
  "issuanceDate": "2010-01-01",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science in Mechanical Engineering"
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-06-18T21:19:10Z",
    "verificationMethod": "https://example.com/jdoe/keys/1",
    "nonce": "c0ae1c8e-c7e7-469f-b252-86e6a0e7387e",
    "signatureValue": "BavEll0/I1zpYw8XNi1bgVg/sCneO4Jugez8RwDg/+
MCRVpjOboDoe4SxxKjkCOvKiCHGDvc4krqi6Z1n0UfqzxGfmatCuFibcC1wps
      PRdW+gGsutPTLzvueMWmFhwYmfIFpbBu95t501+rSLHIEuujM/+PXr9Cky6Ed+W3JT24="
  }
}
```

ACE-OAuth Token Response

Header: Created (Code=2.01)

Content-Format: "application/ace+cbor"

Payload:


```
{
  "access_token" : b64'SlAV32hkKG ...
  (remainder of CWT omitted for brevity;
  CWT contains COSE_Key in the "cnf" claim)',
  "profile" : "coap_dtls",
  "expires_in" : "3600",
  "cnf" : {
    "COSE_Key" : {
      "kty" : "Symmetric",
      "kid" : b64'39Gqlw',
      "k" : b64'hJtXhkV8FJG+Onbc6mxCcQh'
    }
  }
}
```

Backup slide: Implementation

- Used software
  - https://github.com/hyperledger/indy-sdk/
  - https://github.com/bshaffer/oauth2-server-php
- Proofs are processed in JSON using Base64 encoding
- Implementation is written using Python, other bindings are also available for Indy SDK