EBERHARD KARLS UNIVERSITÄT TÜBINGEN

FACULTY OF SCIENCE

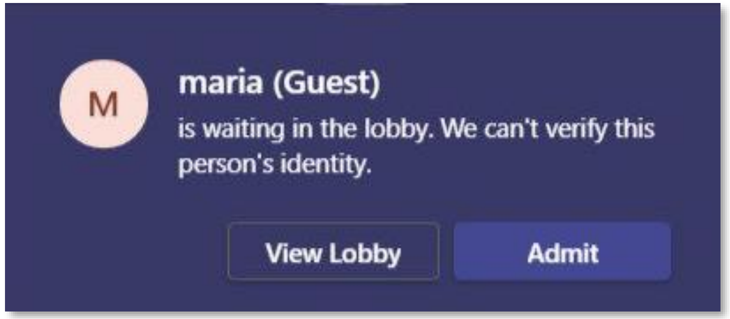
Communication Networks

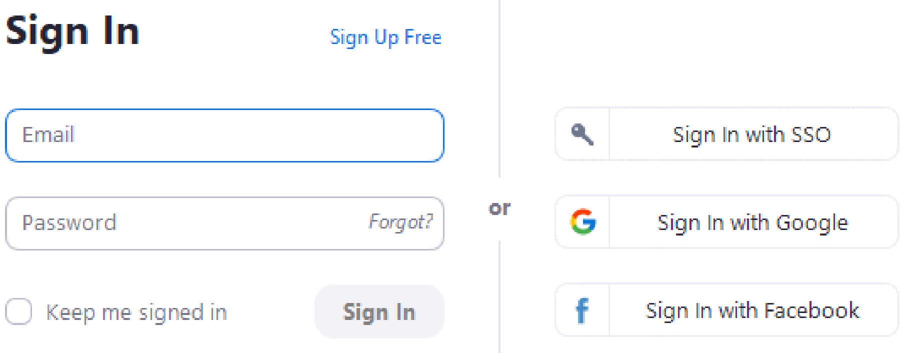
Message Layer Authentication with OpenID Connect

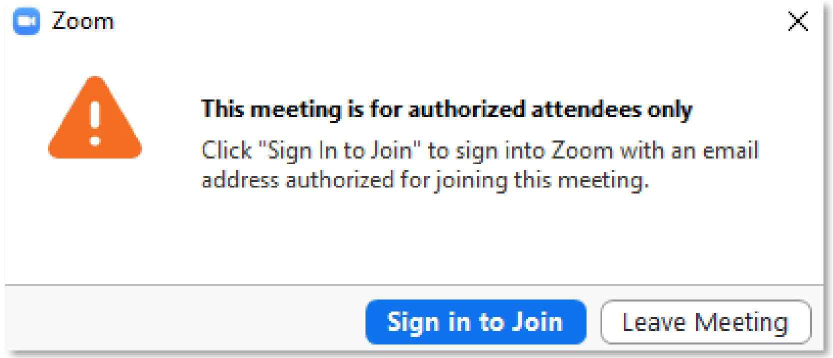
by Jonas Primbs, Chair of Communication Networks, Faculty of Science, University of Tübingen, Germany

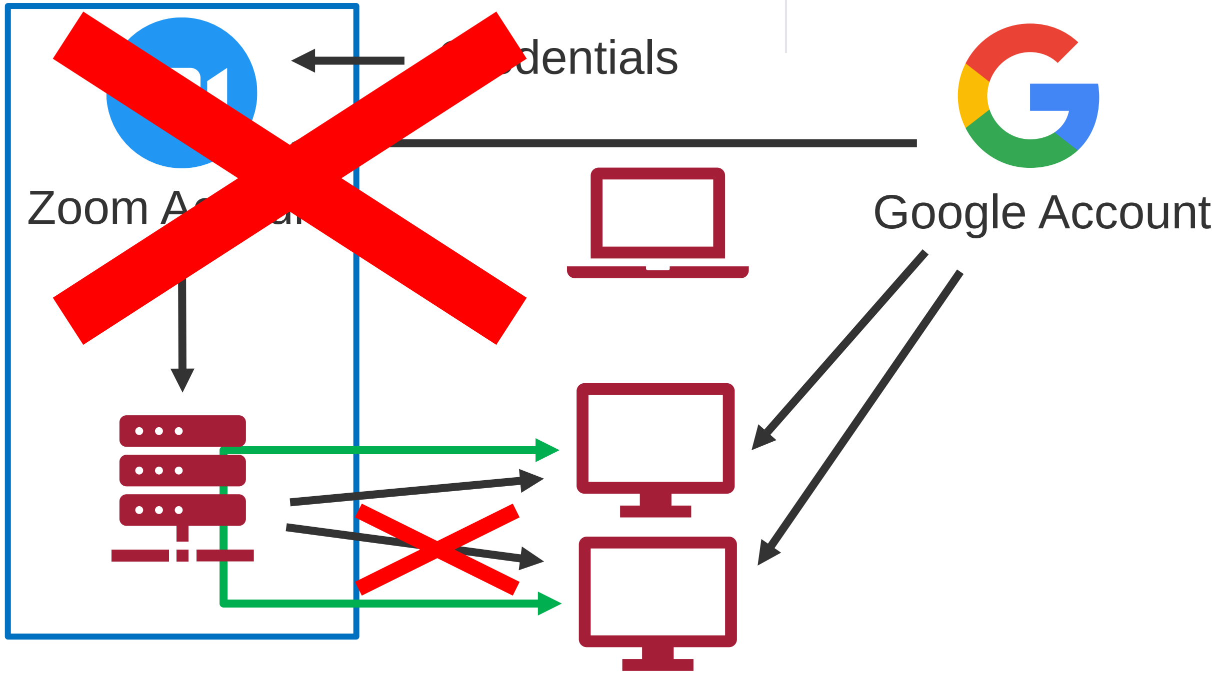
http://kn.inf.uni-tuebingen.de

- Imagine a confidential E2E encrypted video conference …









- We also need E2E authentication!

Outline

- Introduction
- **Outline**
- Clarification
  - What is the Message Layer?
  - Terminology
- Proposed Solution
  - User Authentication
  - ID Assertion Token
  - End-to-End Authentication
  - End-to-End Encryption
- Conclusion
- Discussion
  - Other Use-Cases
  - Open Questions
  - Next Steps

What is the Message Layer?

Message Layer ≥ Application Layer

We are here →

5. **Application Layer: Users send messages**
4. Transport Layer: Services send datagrams
3. Network Layer: Devices send packets
2. Data Link Layer: Devices send frames
1. Physical Layer: Devices send signals

Open System Interconnection (OSI) Model

5. Application Layer
End User Security (OAuth, OIDC, …)

4. Transport Layer
Service Security (TLS)

3. Network Layer
WAN Device Security (IPsec, VPN, …)

2. Data Link Layer
LAN Device Security (MACsec, IEEE 802.1X, …)

1. Physical Layer
Physical Device Security (WPA, …)

Terminology

- We do **authentication** here, **not authorization**
- This presentation is about an **OpenID Connect (OIDC) extension**, not OAuth
  - We use the OIDC terminology

Relying Party (RP) = Client Application

OpenID Provider (OP) = Identity Provider / SSO

End User (EU) = Resource Owner / real person

Identity Claims = Information about a real person (name, email address, profile image, …)

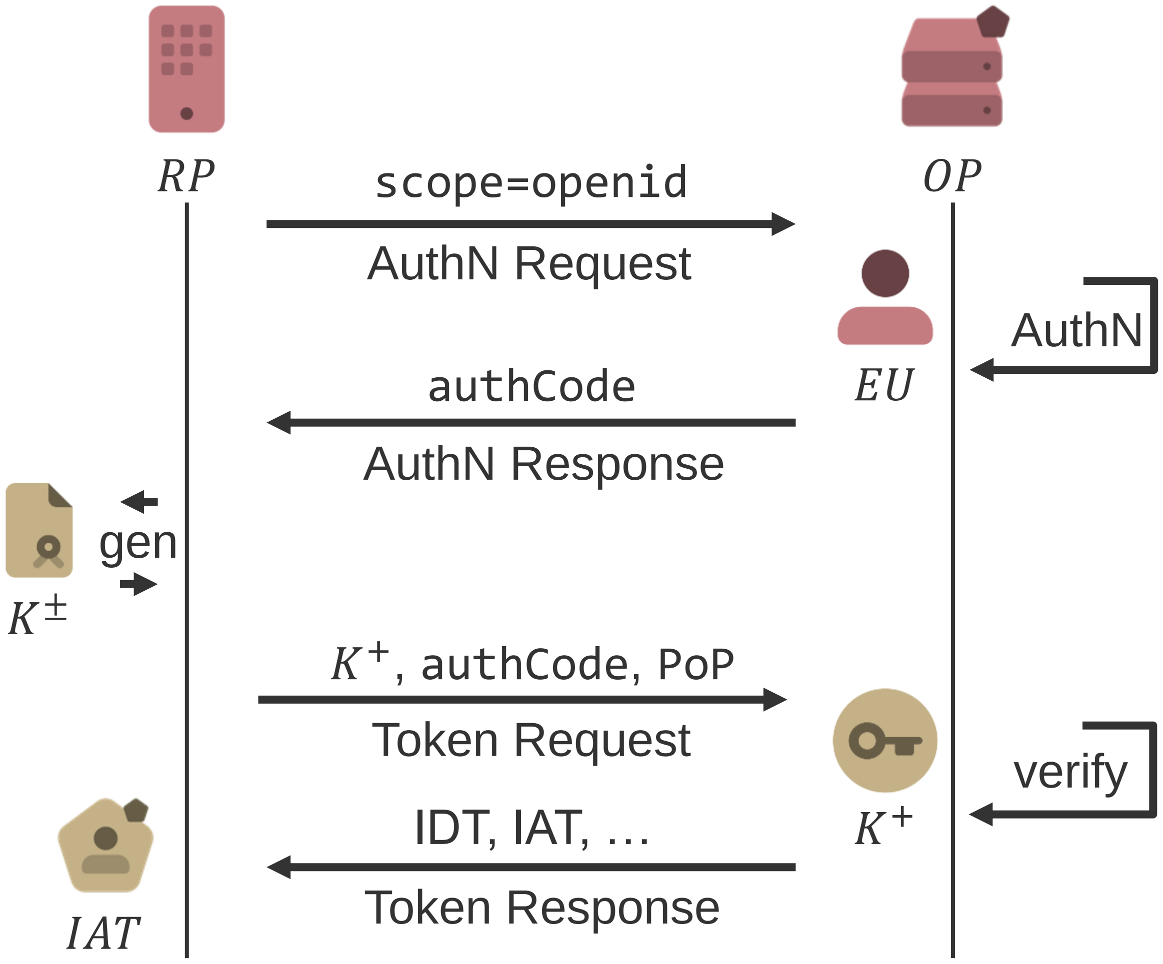
ID Token (IDT) = JSON Web Token (JWT), issued by the OP, which contains Identity Claims

Key Pair $(K^{\pm})$ = Asymmetric public/private key pair / not necessarily signed certificate

Scope = Permissions that an EU grants to its RP

RP requests authentication form EU at OP

1. RP sends Authentication Request to OP
   - MUST contain request for `openid` Scope
2. EU authenticates to OP and grants access to `openid` Scope
3. OP responds with Authorization Code (AC)
4. RP generates asymmetric key pair $K^{\pm}$
5. RP sends Token Request
   - Contains public key K^{+} and AC
   - Contains proof-of-possession of K^{-}
     - DPoP, HTTP Message Signature, mTLS, …
6. OP verifies Token Request
7. RP responds with ID Token (IDT), Access Token (AT), …, and **ID Assertion Token (IAT)**



ID Assertion Token

- **A public sender-constraint ID Token**
  - JSON Web Token (JWT), signed by issuer (= OP)
  - Contains public identity claims
    - name, email address, ~~internal identifiers~~, ~~secrets~~, …
  - Contains the provided and verified public key K^+
    - In confirmation “cnf” claim
    - IAT only valid, if RP proves possession of related private key
  - Expiration date “exp”
    - Avoids revocation list

- Like Verifiable Credential in SSI-world
  - But OP is Root-of-Trust, not a public ledger

- RP shares IAT publicly with other RPs
  - Remote RP must trust issuer (= OP) of IAT
  - IAT is not a secret token

HEADER: ALGORITHM & TOKEN TYPE

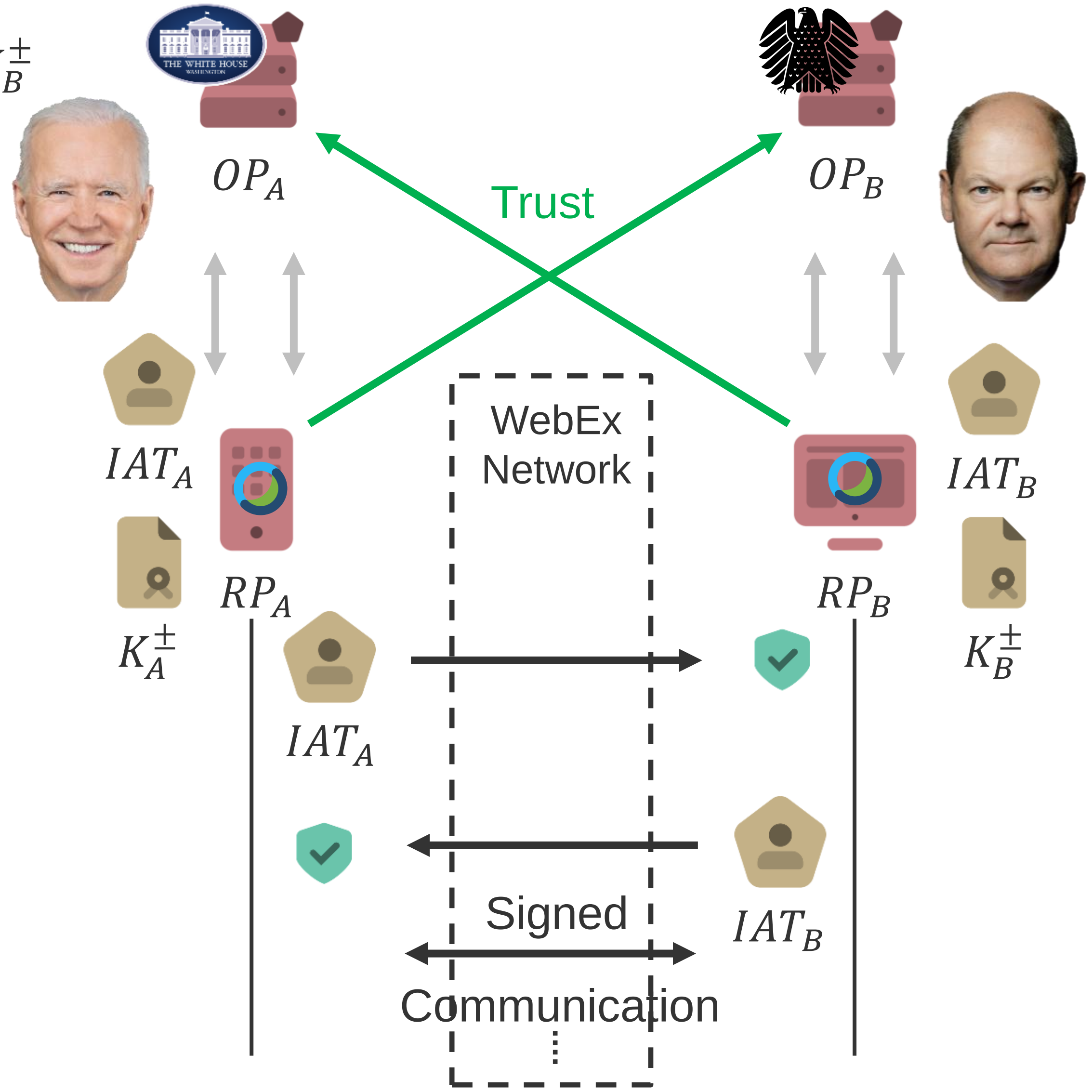

```
{
  "alg": "RS256",
  "kid": "cec13debf4b96479683736205082466c14797bd0",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "iss": "https://accounts.google.com",
  "email": "j.primbs@gmail.com",
  "email_verified": true,
  "name": "Jonas Primbs",
  "given_name": "Jonas",
  "family_name": "Primbs",
  "locale": "de",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  },
  "iat": 1649259849,
  "exp": 1649263449
}
```

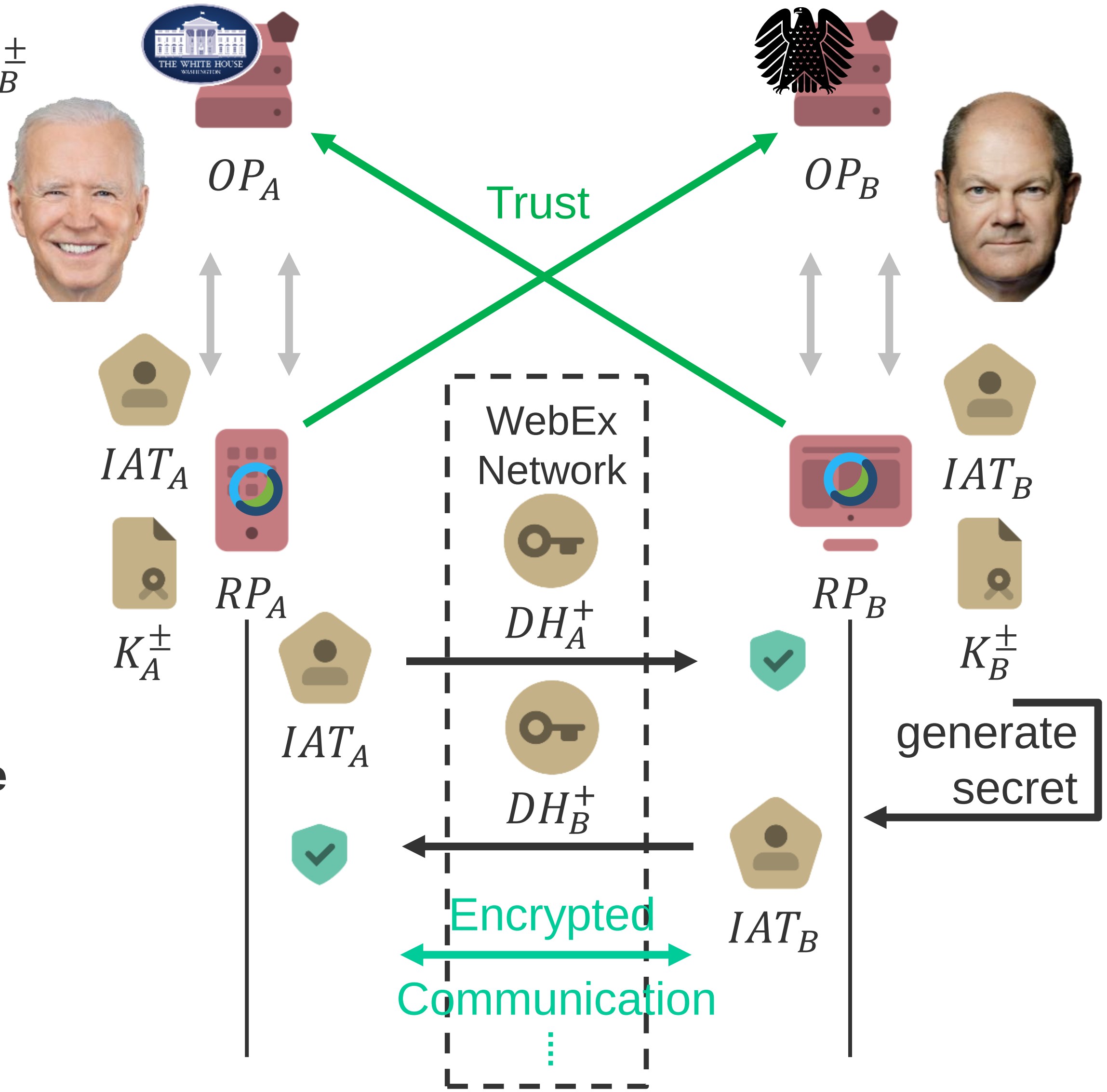
End-to-End Authentication

1. RP A and B obtain IAT A and B, bound to $K_A^{\pm}$ and $K_B^{\pm}$
2. RP A sends IAT A to RP B } Not specified
   - RP A proves possession of K_A^{-}
3. RP B verifies IAT A
   - Requires EU to trust to OP A
   - Verify validity of IAT A (signature, lifetime, …)
   - Verify proof-of-possession
4. RP B sends IAT B to RP A } Not specified
5. RP A verifies IAT A

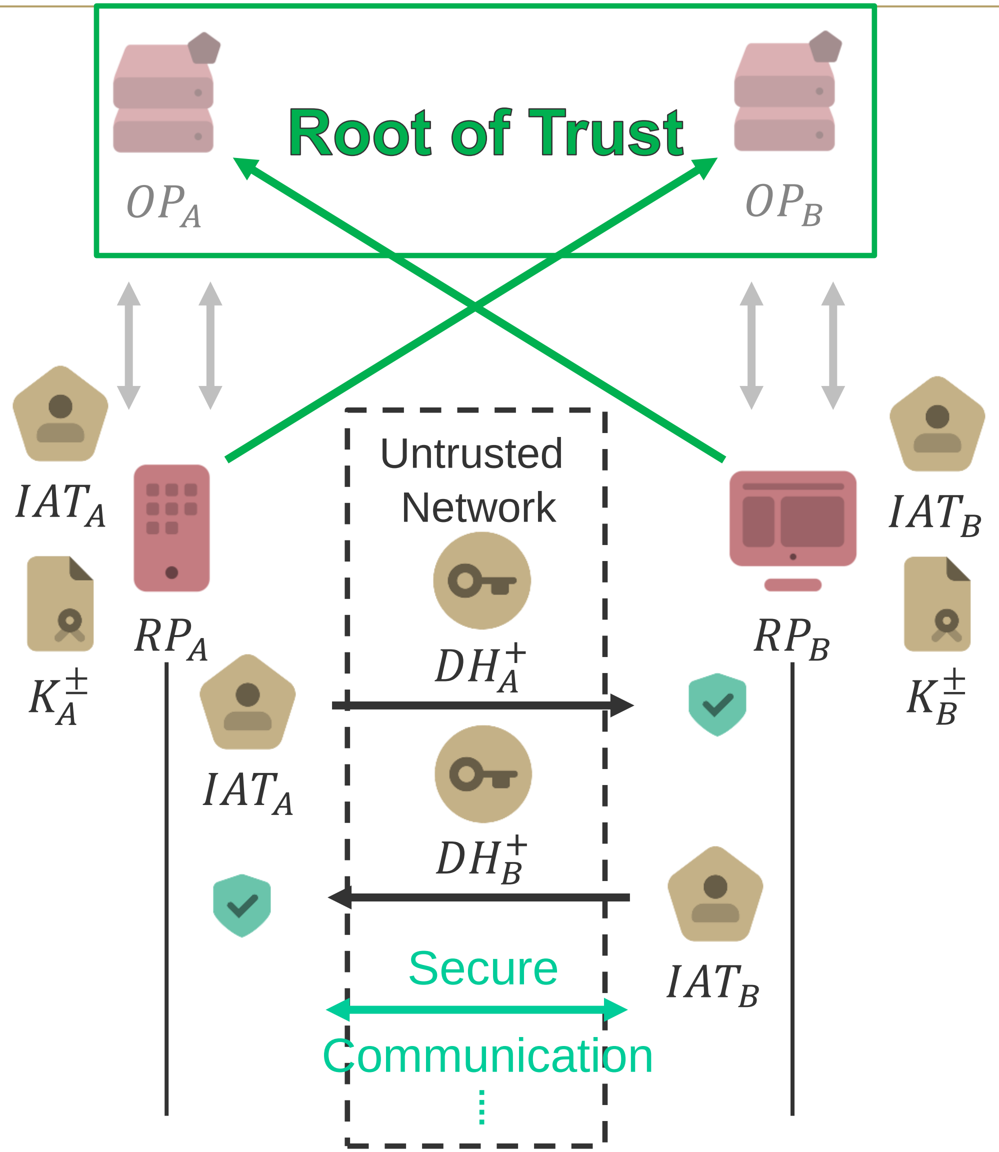


Optional Extension: End-to-End Encryption

1. RP A and B obtain IAT A and B, bound to $K_A^{\pm}$ and $K_B^{\pm}$
2. RP A sends IAT A to RP B
   - RP A proves possession of K_A^{-}
   - **RP A provides signed Diffie-Hellman parameters**
3. RP B verifies IAT A
   - Requires EU to trust to OP A
   - Verify validity of IAT A (signature, lifetime, …)
   - Verify proof-of-possession
   - **RP B generates DH params + DH secret**
4. RP B sends IAT B to RP A
   - **With signed DH params + 1st encrypted message**
5. RP A verifies IAT A
   - **RP A computes shared secret and decrypts encrypted message**



- **New Root-of-Trust**
  - Shifts Root-of-Trust from application to any OpenID Provider
- **Cross-domain authentication**
  - Relying Party (RP) decides, which OpenID Provider (OP) to trust
- **OIDC-based "Lightweight E2E Layer-5 TLS"**
  - Elliptic Curve ($K^{\pm}$) authentication + Diffie Hellman (DH^{+}) key exchange
  - OP = Root-of-Trust ≈ Root Certificate Authority
  - Authentication between End Users (EUs) ≈ Authentication between applications / services



Questions? Comments?

Conclusion and Questions

- **ID Assertion Token (IAT)**
  - A public, sender-constraint ID Token
  - Might be adaptable to Verifiable Credentials

  - **Do we need a subject "sub" claim?**
    - Contains **internal** account identifier at OP
    - subject + issuer = globally unique

  - **Long- or short-term validity?**
    - Short-term: No revocation list required
    - Long-term: Certificate may be X.509 certificate (mTLS, S/MIME, or roaming authenticators like FIDO2 sticks)

Questions? Comments?

HEADER: ALGORITHM & TOKEN TYPE

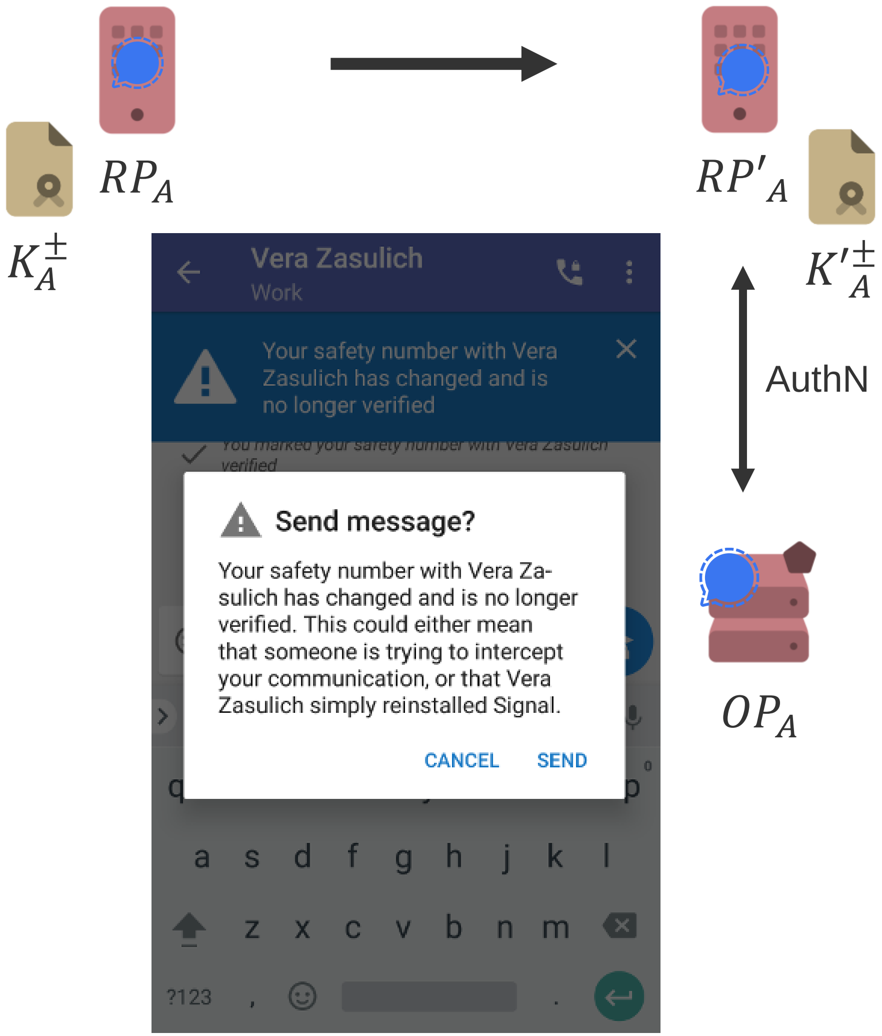

```
{
  "alg": "RS256",
  "kid": "cec13debf4b96479683736205082466c14797bd0",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "iss": "https://accounts.google.com",
  "email": "j.primbs@gmail.com",
  "email_verified": true,
  "name": "Jonas Primbs",
  "given_name": "Jonas",
  "family_name": "Primbs",
  "locale": "de",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "use": "sig",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  },
  "iat": 1649259849,
  "exp": 1649263449
}
```

Use-Case: E2E-Security for Instant Messengers

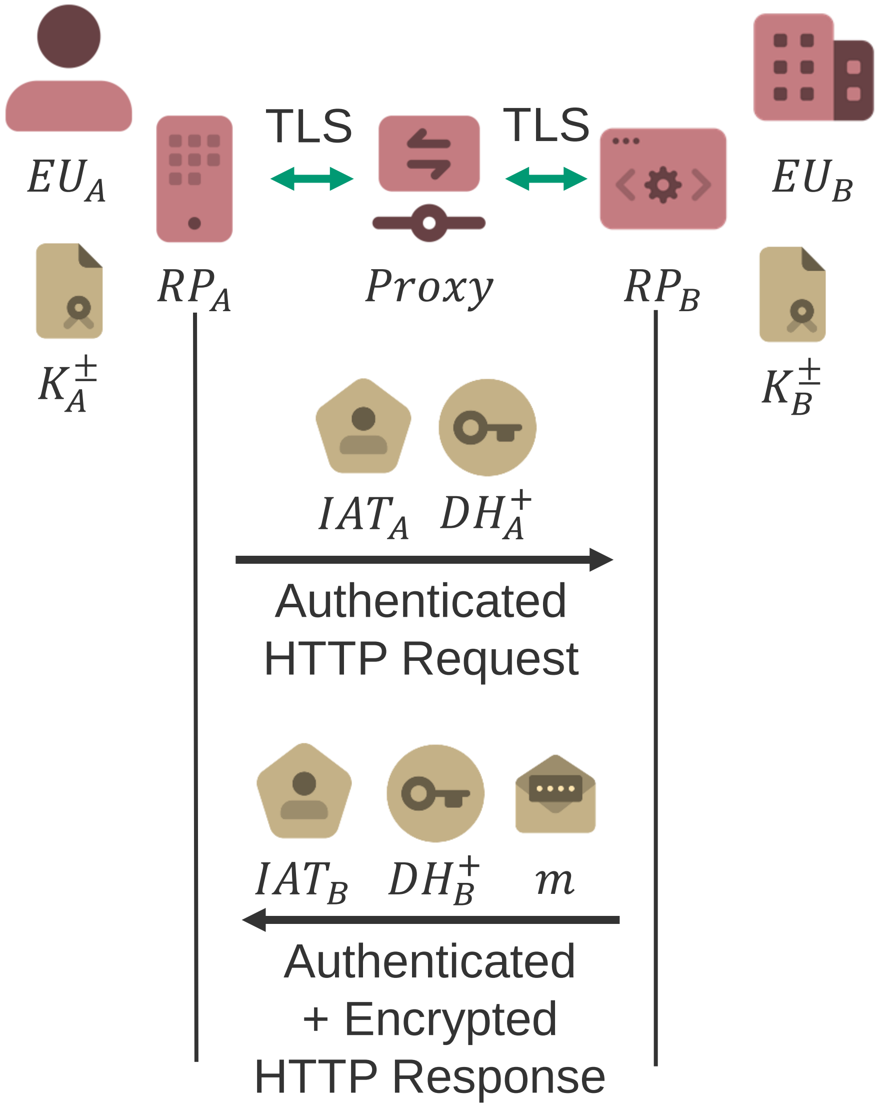
- When moving to another device, the “safety number” (message signing key pair) changes
- Problem: How to deal with changed key pairs?
- Solution: Verify key pair with OpenID Connect!
  - Request ID Assertion Token for key pair from any trustful OpenID Provider
    - Not necessarily the IM provider’s OP
  - Authenticate via SMS verification
  - Include verified phone number as only identity claim in ID Assertion Token
  - Issue IAT and PoP to remote RP



Use-Case: E2E-Security for HTTP

- TLS terminates at each endpoint (e.g., Proxy)
- Authentication mechanism works also on stateless protocols, e.g., HTTP
  - But encryption only for one direction

- Navigator (RP A) requests data from Server (RP B) via HTTP Proxy
  - Proxy can introspect and mutate every message in clear-text
  - Navigator provides IAT A, which is signed by an OP that the Server trusts, and DH params
- Server responds via HTTP Proxy
  - Server provides IAT B, which is signed by an OP that the Navigator trusts and contains an expected Server-specific subject, DH params, and a message m, encrypted with DH secret
  - Navigator generates DH secret and decrypts m



Discussion

Open Questions

- Solution for asynchronous messages like email?
  - Mechanism may also be used as alternative for S/MIME or OpenPGP
- Alternative for SSO Authentication Flow?
  - Direct authentication with IAT against Authorization Server
- GNAP compatibility?
- Other questions and feedback welcome!
  - Via Email: jonas.primbs@uni-tuebingen.de
  - Via GitHub: /JonasPrimbs/draft-ietf-mla-oidc
    - Contains the latest draft

Next Steps

- Where to publish? IETF or OpenID draft?
- Co-Authors: Any volunteers?
- Protocol security analysis

Thank you for listening!

ID Assertion Token Configuration

- Identity Claims in ID Assertion Token are configurable using the optional claims parameter in the Authentication Request

- Privacy first!
  - No identity claims by default = **anonymous** authentication as user in OP domain
  - Identity claims not requested in the `id_assertion_token` object, will be not present in the issued ID Assertion Token!

```
{
  "id_token": {
    "given_name": null,
    "email": {"essential": true}
  },
  "id_assertion_token": {
    "given_name": {"essential": true},
    "family_name": {"essential": true},
    "nickname": null
  }
}
```

Must be available

Present if available

Use-Case: E2E-Security for Instant Messengers

- The European Union plans to enforce interoperability of instant messengers
  - Authentication is no problem **inside** the WhatsApp / Signal / … domain, but across domains!

- Solution
  - Each messenger serves its own OpenID Provider
    - Authentication via SMS verification code
  - E.g., Signal user decides on its client to trust any 3rd party (e.g., WhatsApp) End User