# Designated Verifier Signatures

[Formerly known as ECDH-MAC based signatures]

*Paul Bastian, Micha Kraus*

# Introduction

- Within the context of eIDAS 2.0 and the Verifiable Credentials ecosystem, mainly regular signatures such as ECDSA are used
  - Issuer-Signed Verifiable Credentials / mdocs for data authenticity, e.g. SD-JWT & ISO mdoc “issuerAuth”
  - Signatures for presentations / key binding, e.g. SD-JWT’s KB(key binding)

- Discussions around privacy and (non-)repudiation / deniability
  - German architecture proposal discusses several PID options (Person Identification Data - equivalent to eID cards)
  - “Authenticated Channel” based on ECDH-MAC aims for repudiable presentations
  - Other applications like Hierarchical Deterministic Keys (HDK) also express desire for repudiation

# Repudiation

- Repudiation (or “plausible deniability”) refers to the property that one of the entities involved in an identification transaction can plausibly deny to a third party (i.e., a party not involved in the transaction) in having participated in the transaction after its completion, or can plausibly deny to a third party having provided certain data. The ability to deny the transaction towards third parties does not impact the reliability of the transaction towards the Relying Party involved. Below, two repudiation variants are considered.
    - Deniability of Data Authenticity
    - User deniability

# Arguing on Repudiation

- In case of data breaches with authenticated PID involved, plausible deniability to the public would become highly favourable to the persons affected, especially if the involved Relying Party could arouse social discomfort
- Regarding long-term storage, data leaks are generally more a matter of time rather than probability, since this risk cannot be thoroughly eliminated. Data thieves would obviously have a preference for guaranteed genuine copies over unauthenticated PID.
- In some use cases, where records need to be kept by law, a long-living, non-repudiable declaration of intent is required to be verified by some third party [...] these declarations differ from identifications, so other means like QES are likely to be more appropriate

# Cryptography

- Established cryptographic schemes, e.g. used in
  - ISO/IEC 18013-5: Mobile driving license (mDL)
  - TR-03110: Extended Access Control (German eID Card)

| | | |
|---|---|---|
| 01 | **negotiate a shared key** | • Diffie-Hellman Key agreement<br>• Key Derivation Function<br>• => shared key between signer and verifier |
| 02 | **sign data** | • Message Authentication Code over data using shared key by signer |
| 03 | **verify signature** | • Verifier checks MAC<br>• Reasoning: “If the verifier has not signed the data, the signer must have done so.” |

# Proposal

- DVS algorithms consist of three components
  - a Diffie-Hellman Key Agreement (DHKA)
  - a Key Derivation Function (KDF)
  - a Message Authentication Code algorithm (MAC)
- Algorithm identifiers follow the pattern: `DVS-<DHKA>-<KDF>-<MAC>`.
- We define the JOSE headers necessary for this
  - “jwk” for Signing Party (reusing existing parameter)
  - “rpk” for Verifying Party (new parameter)
  - “nonce” for additional possibility for provide freshness by Verifying Party
- Additionally, we describe algorithms that directly utilize HPKE (RFC 9180)
  - Don’t do encryption, but use underlying auth mode of AEAD

# Example with JWT

The JWT/JWS header:

```
{
    "typ" : "JWT",
    "alg" : "DVS-P256-SHA256-HS256",
    "jwk" : <JWK of the Signing Party>,
    "rpk" : <JWK of Verifying Party>
}
```

The JWT/JWS payload:

```
{
    "iss" : "https://example.as.com",
    "iat" : "1701870613",
    "given_name" : "Erika",
    "family_name" : "Mustermann"
}
```

The JWT/JWS signature:

```
base64-encoded MAC
```

# Algorithm Identifiers

| Algorithm Name | Algorithm Description | Requirements |
|---|---|---|
| DVS-P256-SHA256-HS256 | ECDH using NIST P-256, HKDF using SHA-256 and HMAC using SHA-256 | Optional |
| DVS-HPKE-Auth-X25519 -SHA256 -ChaCha20Poly1305 | DVS based on HPKE using DHKEM(X25519, HKDF-SHA256) HKDF-SHA256 KDF and ChaCha20Poly1305 AEAD | Optional (Appendix A) |
| DVS-HPKE-Auth-P256 -SHA256-AES128GCM | DVS based on HPKE using DHKEM(P-256, HKDF-SHA256) HKDF-SHA256 KDF and AES-128-GCM AEAD | Optional (Appendix A) |

# Summary

- Offers repudiable signatures with established cryptography
- Brings similar mechanism that ISO mdoc offers to (SD-)JWT
- Enables repudiable signatures for data authenticity or user binding
    - however, the RP usually needs to send ephemeral public key (with the correct curve) which results in some challenges for the protocol design

# Questions?

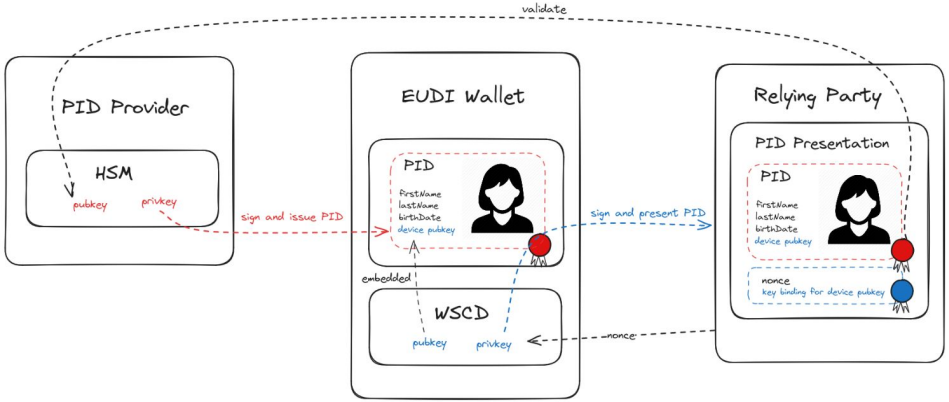
# Links

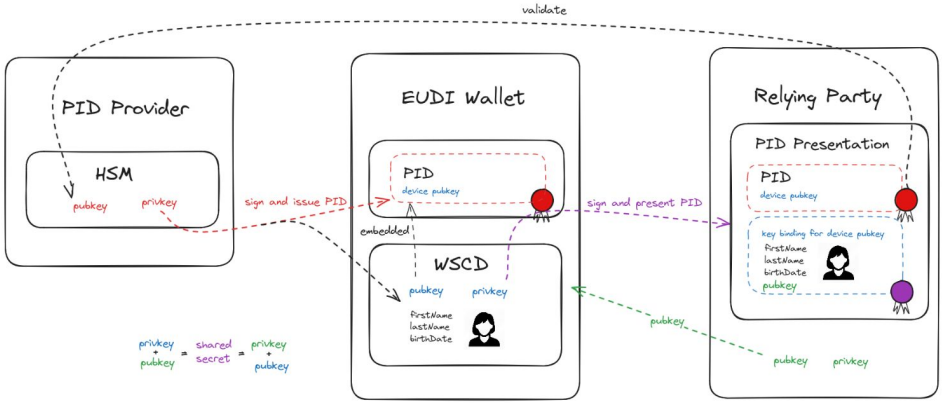
Datatracker -> https://datatracker.ietf.org/doc/draft-bastian-jose-dvs

Git Repository -> https://github.com/paulbastian/draft-bastian-jose-dvs

Current Editors Copy -> https://paulbastian.github.io/draft-bastian-jose-dvs/draft-bastian-jose-dvs.html

# Backup


validate
PID Provider
HSM
pubkey
privkey
sign and issue PID
EUDI Wallet
PID
FirstName
lastName
birthDate
device pubkey
embedded
WSCD
pubkey
privkey
sign and present PID
Relying Party
PID Presentation
PID
FirstName
lastName
birthDate
device pubkey
nonce
key binding for device pubkey
nonce


validate
PID Provider
HSM
pubkey
privkey
sign and issue PID
EUDI Wallet
PID
device pubkey
embedded
WSCD
pubkey
privkey
FirstName
lastName
birthDate
sign and present PID
Relying Party
PID Presentation
PID
device pubkey
key binding for device pubkey
FirstName
lastName
birthDate
pubkey
pubkey
privkey
privkey
pubkey
shared
secret
privkey
pubkey