ECDH-MAC based signatures

A simple and scalable credential revocation/status mechanism
[Formerly known as JWT CWT Status List]

Paul Bastian

Introduction

- German national ID card (launched in 2010) has online functionality for identification/pseudonymous authentication using Extended Access Control (EAC) protocols
  - Protocols do not send signed data or leak other cryptographic identifiable data, privacy-by-design
  - Using ECDH key agreement and sending data over symmetric channel in plain
  - Additionally using group keys, similar to what FIDO did later
- This sets high bar in terms of privacy for eIDAS 2.0-based protocols

Motivation

Within the architecture team for the german eIDAS Wallet, we investigate various wallet architecture designs. Besides other factors, one of the properties that is being analysed and compared is:

- Issuer-signed Credentials vs Authenticated Channel Credentials

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 thiefs 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

Proposal

- JOSE algorithm that performs MAC based on derived key from ECDH key agreement
  - Acting more like a signature scheme
- Respecting the ideas of Fully-Specified Algorithms, 3 parameters:
  - the domain parameters for the ECDH "curve"
  - the key derivation algorithm "kd"
  - the MAC algorithm "mac"
- Example alg name: “ECDH-P256-CC-HS256”
-

Algorithm

- The generation of ECDH-based MAC follows these steps:
  - Perform ECDH as defined as defined by "curve" - use the specified elliptic curve to generate a key pair and set the epk - use the Verifier's public key defined by kid to perform the key agreement - optionally provide a certificate chain defined by x5c
  - Derive symmetric key as defined by "kd" - use the output from the key agreement as an input for the key derivation algorithm - derive the MAC key
  - Generate a MAC as defined by "mac" - use the output from the key derivation algorithm as an input for the MAC algorithm - generate the MAC
- The verification of ECDH-based MAC follows these steps:
  - Perform ECDH as defined as defined by "curve" - use the specified elliptic curve to generate an ephemeral key pair and set the kid - provide the public key kid to the Generating Party - use the Generating Party's public key defined by epk and perform the key agreement - optionally validate the certificate chain defined by x5c
  - Derive symmetric key as defined by "kd" - use the output from the key agreement as an input for the key derivation algorithm - derive the MAC key
  - Generate a MAC as defined by "mac" - use the output from the key derivation algorithm as an input for the MAC algorithm - generate the MAC - compare the generated MAC with the signature value

Example JWT

```
{
  "alg": "ECDH-P256-HS256",
  "x5c" : <issuer certificate chain that signs the epk>,
  "epk" : <JWK used for ECDH>,
  "kid" : <key ID of Verifying Party>
}
.
{
  "iss": "https://example.com",
  "given_name" : "Erika",
  "family_name" : "Mustermann"
}
.
{
   base64-encoded MAC
}
```

Summary

- Offers repudiable signatures with established crypto
- Brings similar mechanism that mdoc offers to (SD-)JWT
- Enables repudiable, privacy-preserving On-Demand/On-the-fly Issuance/Presentation of VC
    - The Issuer does not even get to know the RP as Wallet is the intermediary
- Enables repudiable, privacy-preserving VC presentation from a Wallet with Trusted Logic (e.g. Javacard, similar to EAC)
- As a disadvantage, 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-ietf-oauth-status-list/

Git Repository -> https://github.com/oauth-wg/draft-ietf-oauth-status-list

Current Editors Copy -> https://oauth-wg.github.io/oauth-sd-jwt-vc/#go.draft-ietf-oauth-sd-jwt-vc.html