# Combining eIDAS High device binding with unlinkability

**By Sietse Ringers, mail@sietseringers.net**

The eIDAS regulation requires SSI wallets that aim to achieve level High to have strong device binding, using a private key from the SE/TEE on which the wallet is running. In practice this requires the usage of ECDSA, since that is the only signature scheme supported by all contemporary SE/TEE's. On the other hand, it is desirable for an SSI wallet to support unlinkability: when identical data is disclosed to a verifier during two authentication sessions, for example the fact that the user is over 18, then it should be infeasible for the verifier to tell if the two authentication session originated from a single user who performed two sessions, or from two distinct users. These two requirements are difficult to combine, since the usage of an ECDSA private key always creates linkability.

The IRMA project, an Idemix implementation used in production in the Netherlands, solves this issue at the cost of having a TTP called the keyshare server, whose primary objective is to confine the linkability due to ECDSA entirely towards itself. The TTP is designed such that it is required to contribute to every authentication session of every user, while learning as little as possible about the session: it recognizes the user and authenticates her using ECDSA, but it does not know to whom the user is disclosing attributes (i.e., data signed by the issuer, or attestations), or which attributes are being disclosed, or their values. In more detail, this works as follows.

- When a user starts her IRMA app for the first time, the app registers its ECDSA public key at the TTP.
- During issuance of an Idemix credential, the IRMA issuance protocol enforces that the user's *secret key* - that is, the first attribute of the credential, whose random value is shared by all credentials of the user - becomes the sum of one part known exclusively by the user, and another part known exclusively by the TTP.
- During disclosure sessions, the user and TTP perform a multiparty computation to create a zero-knowledge proof over the secret key. Since neither party knows the complete secret key, the only way that such a zero-knowledge proof can be created is by computing it together. The TTP communicates exclusively with the user in this protocol, which prevents the TTP from learning the identity of the verifier to which the user is disclosing attributes. Additionally, the protocol is such that the TTP is exclusively involved with the secret key and none of the other attributes from the user's credential.
- The TTP will only engage in that multiparty computation during disclosure if the user succesfully authenticated herself using ECDSA towards the TTP.

At the end of the protocol, the user sends an ordinary Idemix disclosure of attributes towards the verifier. Since Idemix by itself supports unlinkability, and since this protocol confines the linkability due to ECDSA entirely to the TTP, this mechanism retains unlinkability towards the verifier while still supporting ECDSA authentication using a private key from the device's SE/TEE. It does, however, come with the following tradeoff:

- The necessity to have the TTP, which needs to be trusted to correctly perform the ECDSA authentication of the user. Ideally such a component would of course not be required, but we remark that in terms of the trust model, this situation is still better than currently used authentication mechanisms such as "Login with Google/Facebook/Apple". In such systems, Google/Facebook/Apple essentially serves as the TTP for the entire authentication process including the resulting attestations about the user, instead of just the verification of the device binding using ECDSA.
- Since the TTP's involvement is required during every single authentication session, the user always needs to have an internet connection when she wants to use her attributes: offline usage of attributes is not supported.

Depending on the use cases and the preferences of all parties involved, this may be an acceptable tradeoff. It would however be interesting if there are other ways than this one to retain unlinkability while simultaneously having sufficiently high device binding.