HSM-assisted EUDI-wallet
&
Qualified Remote Signing
based on Split-ECDSA (SECDA)

Eric Verheul

Agenda

- SECDSA features
- The foundation: Split-ECDSA (SECDSA)
- HSM assisted EUDI-wallet based on standard mobile hardware
- Proof-of-Associations on standard mobile hardware
- Qualified Remote Signing (sketch)

SECDSA features

- Allows HSM assisted EUDI-wallet based on standard mobile hardware (*) with the following features:
  - Optimal security (no information stored in wallet or stored/processed at WP allowing for PIN brute-force)
  - Support for publicly verifiable, non-reputable signatures on wallet instructions providing:
    - provable “sole control” and transaction transparency,
    - expedient dispute resolution for users,
    - liability reduction for wallet provider and (PID) issuers.
  - Can be based on HSM PKCS#11 standard.
  - Efficient as requires only one HSM PKCS#11 call (DH) overhead per wallet authentication.
- Allows Proof-of-Association for standalone EUDI-wallet using standard mobile cryptographic hardware (*).
- Same approach can be used in the context of remote signing conform EN 419241-2 (SAM) and EN 419221-5 (Cryptographic Module for Trust Services) providing qualified remote signing with provable “sole control”.

(*) iOS/Secure Enclave, Android/Hardware Backed Keystore or StrongBox, Windows/TPM.

The foundation: Split-ECDSA (SECDSA)

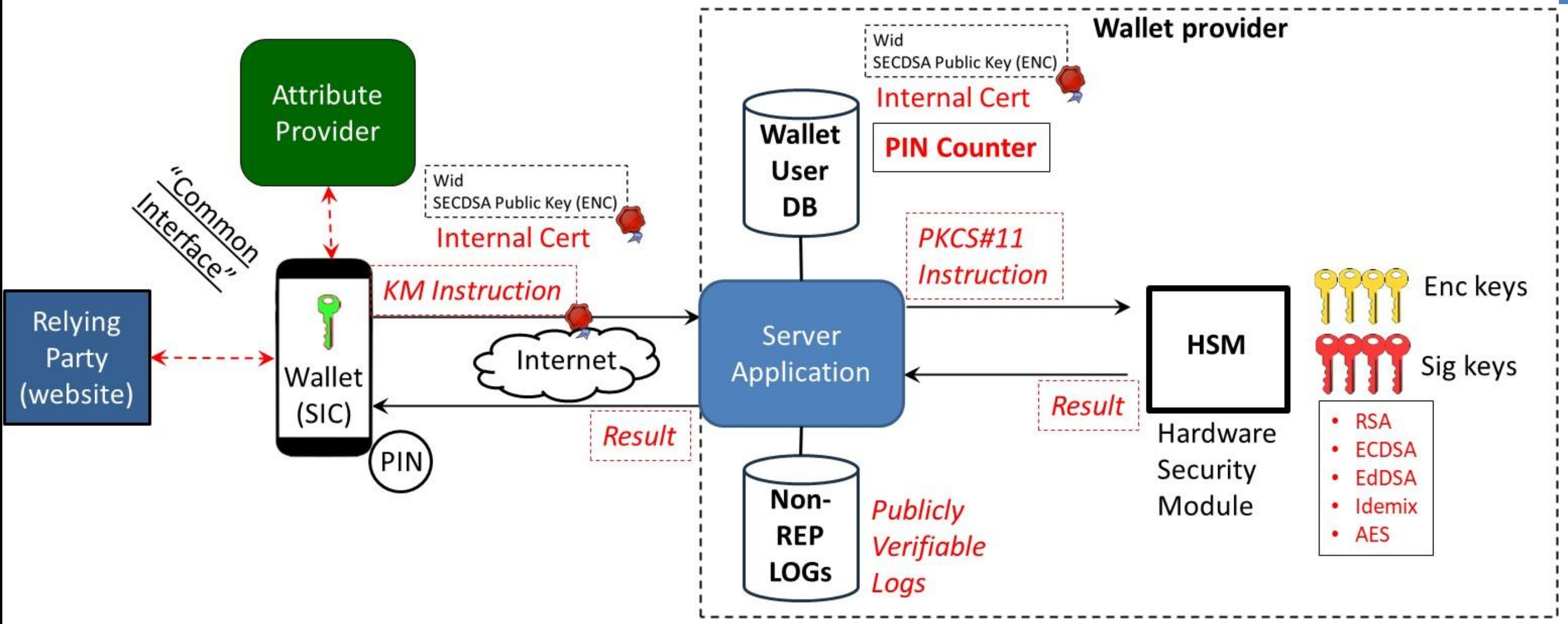
Algorithm 6 Split-ECDSA (SECDSA) signature generation
Input: message M, PIN-key $\sigma \in \mathbb{F}_q^*$, SCE-key $u \in \mathbb{F}_q^*$
Output: ECDSA signature (r, s) corresponding to public key $\sigma \cdot u \cdot G$.

1: Compute $\mathcal{H}(M)$ and convert this to an integer e.
2: Compute $e' = \sigma^{-1} \cdot e \bmod q$
3: Select random $k \in \{1, ..., q-1\}$
4: Compute $kG = (x, y)$ and convert x to integer $\bar{x}$
5: Compute $r = \bar{x} \bmod q$. If $r = 0$ go to Line 1
6: If $r \bmod q = 0$ then go to Line 1
7: Compute $s_0 = k^{-1}(e' + u \cdot r) \bmod q$. If $s_0 = 0$ go to Line 1
8: Compute $s = \sigma \cdot s_0 \bmod q$
9: Return (r, s)

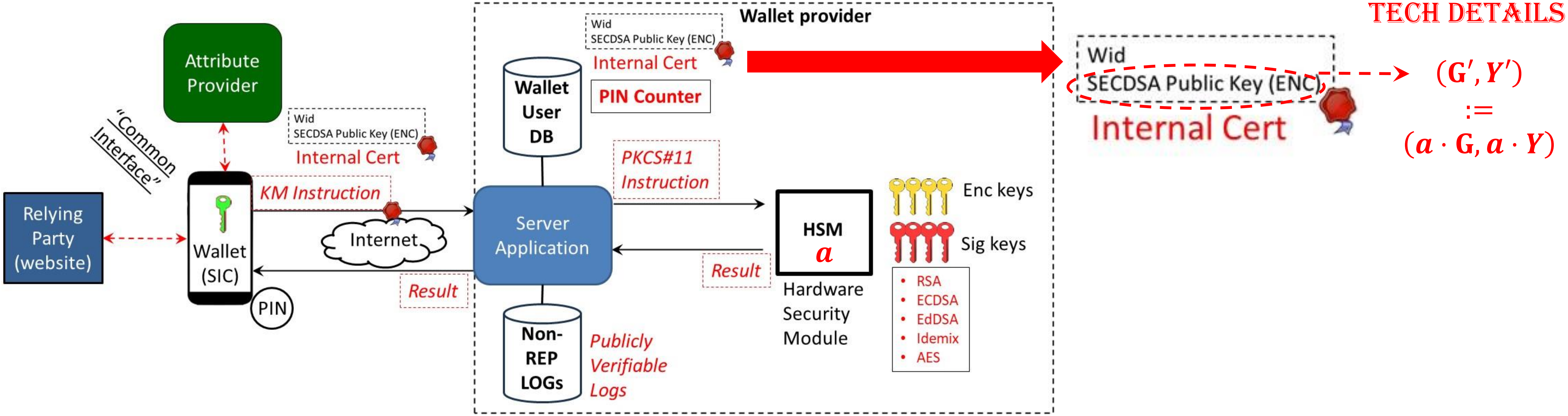
Call to SCE (=hardware) [lines 3–7]

- The mobile cryptographic hardware is called Secure Cryptographic Environment (SCE) in the SECDSA paper.
- The PIN-key σ is derived from the user PIN and another key in the SCE: each PIN results in a different PIN-key.
- The public key $\mathbf{Y} = \sigma \cdot \mu \cdot G$ and signature (r, s) are called the raw SECDCA public key and raw signature.
- That (r, s) is a correct ECDSA signature for private key $\sigma \cdot \mu$ is a simple verification.
- Raw SECDSA public key/signature allow for PIN brute-force: may not be stored or leave wallet unencrypted.
- By repetitive SCE use (output = input) the generation time of the PIN-key can be controlled, e.g. set to 1 second. This allows controlling the expected PIN-brute-force time and thus the effectiveness of PIN-brute-force.
- The key σ can also be protected by biometric (finger, face) access control of the platform; this could be the base for an eIDAS substantial stand-alone EUDI-wallet.

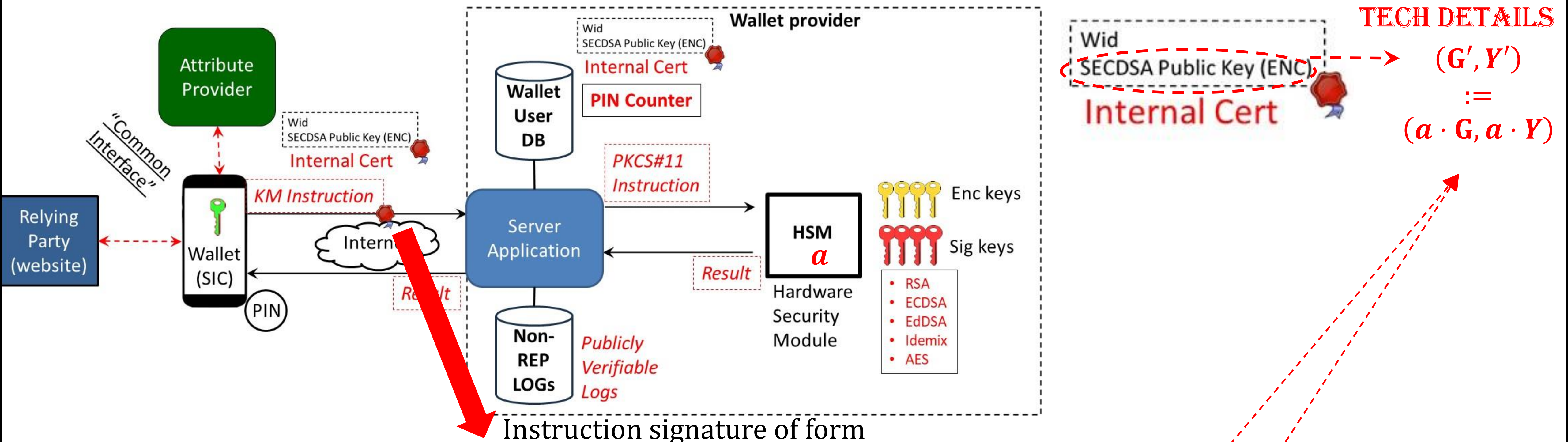
HSM assisted EUDI-wallet based on standard mobile hardware



- During wallet initialisation, an *Internal Certificate* (IC) is agreed between wallet and wallet provider.
- Internal certificate holds unique Wallet Identifier (WId) and homomorphically encrypted raw SECDSA public key with a DH key managed by the WP HSM to prevent PIN brute-forcing. Raw SECDSA Public is not revealed to WP.
- The IC is stored in the Wallet User DB together with a PIN counter.
- SECDSA signatures on Key Management (KM) instructions are also homomorphically encrypted allowing WP verification against encrypted raw SECDSA public key without information appearing allowing PIN brute-force.
- When correct, the SECDSA signatures on the KM instructions are made publicly verifiable by the WP HSM allowing for non-repudiation of the KM instruction.
- All homomorphic encryption techniques are very simple (see next slides).



- Homomorphically encrypted raw public key Y takes form $(a \cdot G, a \cdot Y)$ with secret scalar a managed in HSM.
- By using standard blinding techniques, the WP gets hold of the encrypted raw public key without seeing it.
- In practical implementations, each wallet/user gets its own secret scalar a (Diffie-Hellman key).

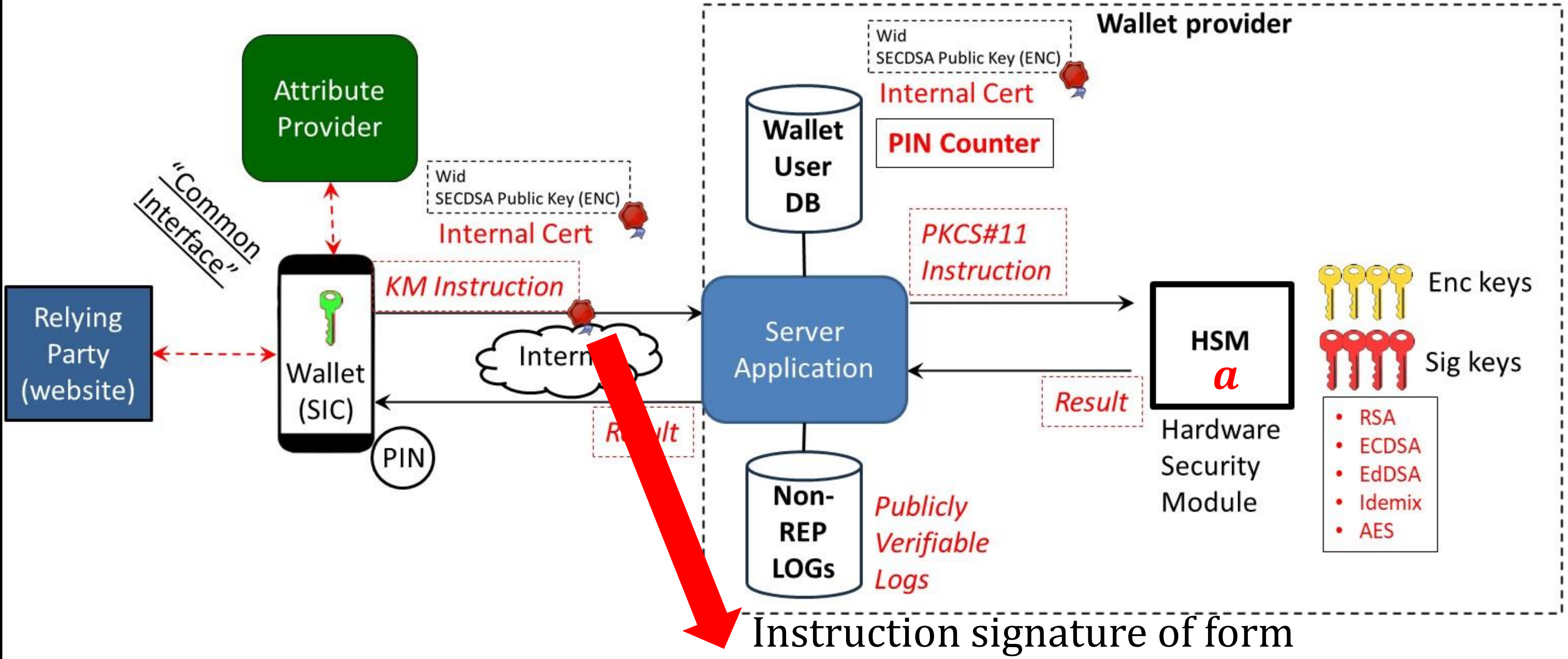


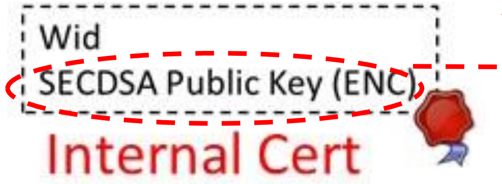
Instruction signature of form

$(U, V, W) \coloneqq (R, s^{-1} \cdot G', s^{-1} \cdot Y')$ plus ZK1

Raw SECDSA signature (r, s) on a Wallet Key Management (KM) instruction is encrypted by the wallet in two steps:

1. It is first transferred into an equivalent form (R, s) with $R \in \langle G \rangle$. Compare Algorithm 3 of SECDSA paper.
2. Signature is homomorphically encrypted as $(U, V, W) \coloneqq (R, s^{-1} \cdot G', s^{-1} \cdot Y')$ plus a Zero-Knowledge proof ZK1, e.g. Schnorr, proving this ($\exists\, x: (V, W) = (x \cdot G', x \cdot Y')$).





$(\mathbf{G}', \boldsymbol{Y}') := (\boldsymbol{a} \cdot \mathbf{G}, \boldsymbol{a} \cdot \boldsymbol{Y})$

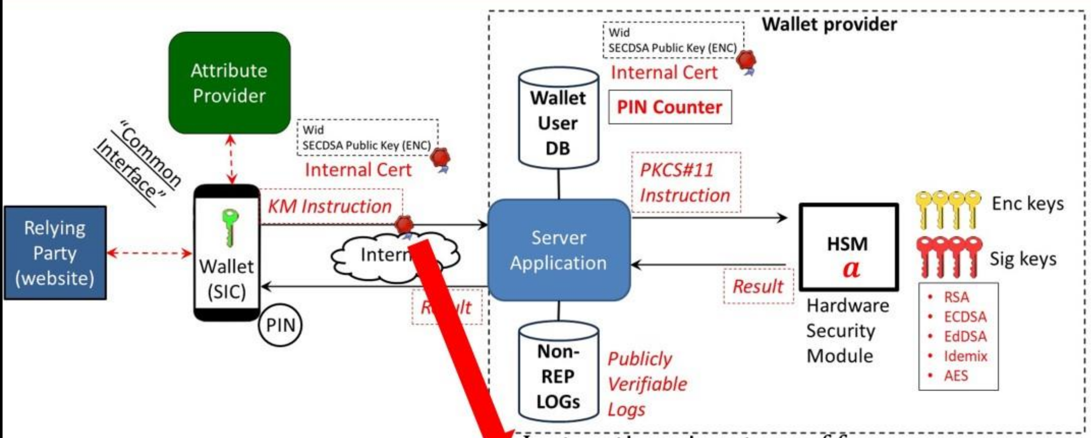
Instruction signature of form

$(U, V, W) := (R, s^{-1} \cdot G', s^{-1} \cdot Y')$ plus ZK1

The encrypted signature is validated by the Wallet provider as follows:

1. The original r is reformed from U (i.e. the originally named R).
2. Homomorphic verification: $a \cdot U \ =^{(*)} \ \mathrm{Hash(M)} \cdot \mathrm{V} + \mathrm{r} \cdot \mathrm{W}$ *// Left side is DH operation*
3. If Step is not successful PIN counter is increased/account blocked etc.…
4. If Step 2 is successful:
   - an extra Schnorr ZK2 is formed making **(*)** publicly verifiable; final SECDSA signature is (U, V, W), ZK1, ZK2.
   - The instruction is performed and the result is returned including the final SECDSA signature.

TECH DETAILS



Instruction signature of form

$(U, V, W) := (R, s^{-1} \cdot G', s^{-1} \cdot Y')$ plus ZK1

TECH DETAILS

Wid
SECDSA Public Key (ENC)
Internal Cert

$(G', Y') := (a \cdot G, a \cdot Y)$

The encrypted signature is validated by the Wallet provider as follows:

1. The original r is reformed from A (i.e. R).
2. Homomorphic verification: $a \cdot U =^{(*)} \mathrm{Hash(M)} \cdot V + r \cdot W$
3. If Step is not successful PIN counter is increased/account blocked etc….
4. If Step 2 is successful:
   - an extra Schnorr ZK2 is formed making **(*)** publicly verifiable; final SECDSA signature is (U, V, W), ZK1, ZK2.
   - The instruction is performed and the result is returned including the final SECDSA signature.

7

r is equal to the x-coordinate of A modulo q the group order.

$\exists\, x: (G', \mathrm{Hash(M)} \cdot L + r \cdot M) = (x \cdot G, x \cdot R)$

(it follows x=a so (*) of previous slide holds)

ZK2 is not time critical, hence can be generated in quiet hours.

Or better: $\mathrm{SHA256}(a \cdot K) =^{(*)} \mathrm{SHA256}(\ \mathrm{Hash(M)} \cdot L + r \cdot M\ )$

Notes:

- $K \rightarrow \mathrm{SHA256}(a \cdot K)$ is DH operation supported by PKCS#11.
- We thus only need one PKCS#11 call to the HSM for the SECDSA signature verification.
- The generation of ZK2 can be done in quiet hours.

09:32 93%

SECDSA POC

Code written by Eric Verheul, all rights reserved.
See https://eprint.iacr.org/2021/910 for SECDSA specification.

ECDSA private key (SCE) hardware backed: true
RSA private key (PIN-Binder) hardware backed: true

Time for 10 PIN-key RSA decryption input iterations is 283 milliseconds, so we use 36 iterations for a 1 second SECDSA signature generation.

**DATA SENT BY APP TO WALLET PROVIDER (WP) FOR ISSUANCE OF
**SECDSA CERT + QUALIFIED PUB KEY
Raw SECDSA key: 020892a36b5e0e5...

**SECDSA CERT QUALIFIED PUB KEY ISSUED BY WP FOR APP/USER
SECDSA based certificate (SF internal)
- Common Name: Eric Verheul
- Encrypted_Pub_Key-G part: 03e8328573ab5d3...
- Encrypted_Pub_Key-Y part: 03e51f4ce244422...
- Certificate Signature: 68dd21935500e38...
Qualified public key (managed in SF HSM):
0212fa15fc7eed82c729efd637bb9ab113fd9e26...

Message to be signed: "Hello World!"

**DATA GENERATED/SENT BY APP TO WP
Message hash value: 86933b0b147ac4c...
User provided SECDSA signature:
- R-part: 03b2c47283d3e42...
- Encrypted s-part1: 03b8e2f3b2b059a...
- Encrypted s-part2: 0246ff21d4f5408...
- Schnorr SF PoK r: 282af3442c2c120...
- Schnorr SF PoK s: 560b7f1e399ae9b...

WP: User provided SECDSA signature is correct!

**DATA GENERATED/SENT BY WP TO APP FOR RP
Qualified signature on message (HSM based private key):
30450220304a309be803e44f0f695e7c...
SECDSA evidence:
- R-part: 03b2c47283d3e42...
- Encrypted s-part1: 03b8e2f3b2b059a...
- Encrypted s-part2: 0246ff21d4f5408...
- Schnorr SF PoK r: 282af3442c2c120...
- Schnorr SF PoK s: 560b7f1e399ae9b...
- Schnorr RP PoK r: 1e8f3a02370eda3...
- Schnorr RP PoK s: 404c1c71e690978...

Qualified signature provided by User/WP to RP correct? true
SECDSA evidence signature provided by User/WP to RP correct? true

Android Studio project available.

G'
Y'
Internal Certificate

R
$s^{-1} \cdot G'$
$s^{-1} \cdot Y'$
ZK1

ZK2
Result of instruction

Note: these operations are performed by Wallet Provider, i.e. not in the APP. POC only.

Proof-of-Associations on standard mobile hardware

1. The *Wallet Trust Attestation* (WTA) is a privacy friendly ISO 23220-3 Secure Area Attestation Object (SAAO)
2. The WTA is an attestation bound to a ECDSA public key $\mathrm{U} = u \cdot G$ whereby the Wallet Provider guarantees:
   a) the wallet/user has possession of u,
   b) u is managed in the wallet SCE (mobile cryptographic hardware).

   Note: a WTA is typically issued by the Wallet Provider based on mobile platform (key) attestation capabilities.
3. The wallet/user can generate a public key V *associated* with the WTA public key U by generating a random scalar k and letting $\mathrm{V} = k \cdot U$. The scalar k could be static, derived from a SCE master key or from a user PIN.

   Note: this fits the SECDSA setup allowing the wallet to ECDSA sign with the private key $v = k \cdot u$.
3. The wallet/user can prove that two public keys U_1, U_2 are associated by proving possession of a private key y: $\mathrm{y} \cdot U_1 = U_2$.

   Notes:
   - *See also this [LinkedIn post](#).*
   - *If the public keys are $U_1 = k_1 \cdot U$, $U_2 = k_2 \cdot U$ are associated then $y = k_2\, k_1^{-1}$.*
   - *The Proof-of-Association (PoA) can be given for instance using a Schnorr Zero-Knowledge Proof ('signature') or alternatively by an ECDSA signature.*
   - *PoAs are verified during issuance by (PID) issuers against the WTA public key, or against another public key that is known to be associated to this WTA key, e.g. a 'WTA copy'.*
   - *If the issuer association verification is registered in the attestations, then relying parties can infer from a PoA that attestations are associated to one WTA (and thus one wallet) and one person.*
   - *PoAs always needs to be accompanied by a proof of possession of the keys involved; efficient combination is possible.*

Qualified Remote Signing (sketch)