EUDI-wallets based on Split-ECDSA (SECDA) and EUDI-wallet roadmap

Eric Verheul

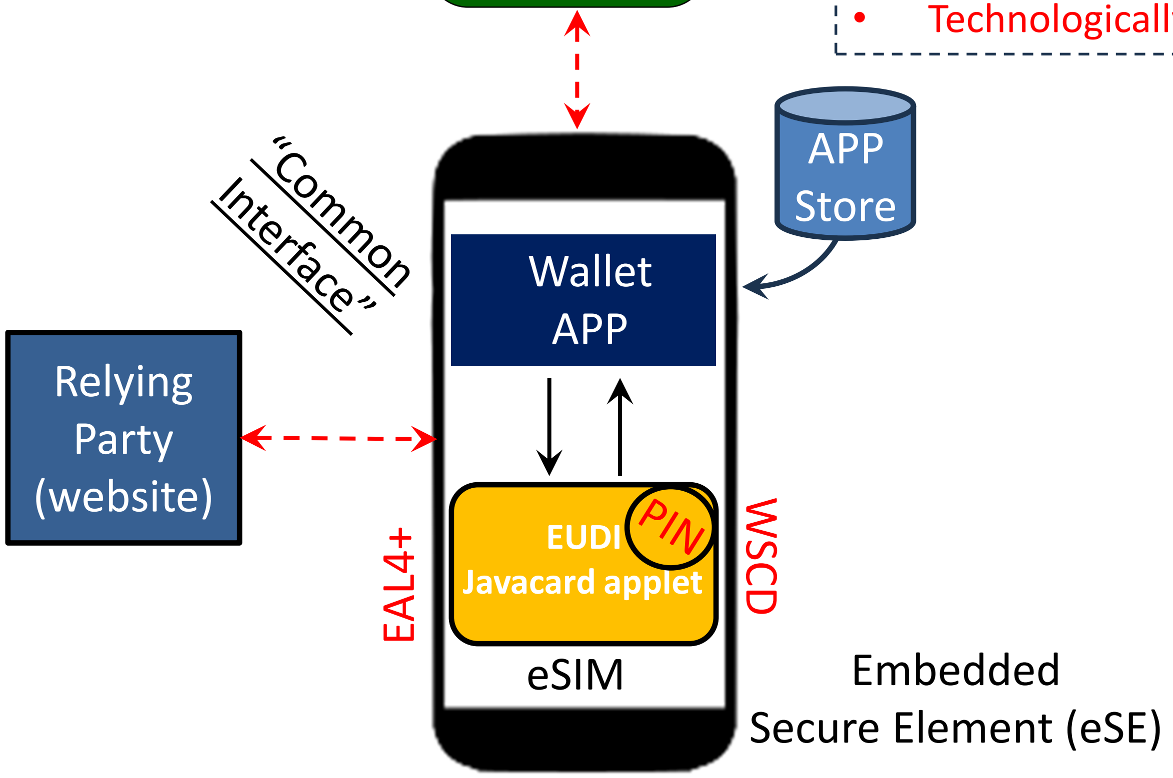
All views expressed are personal only

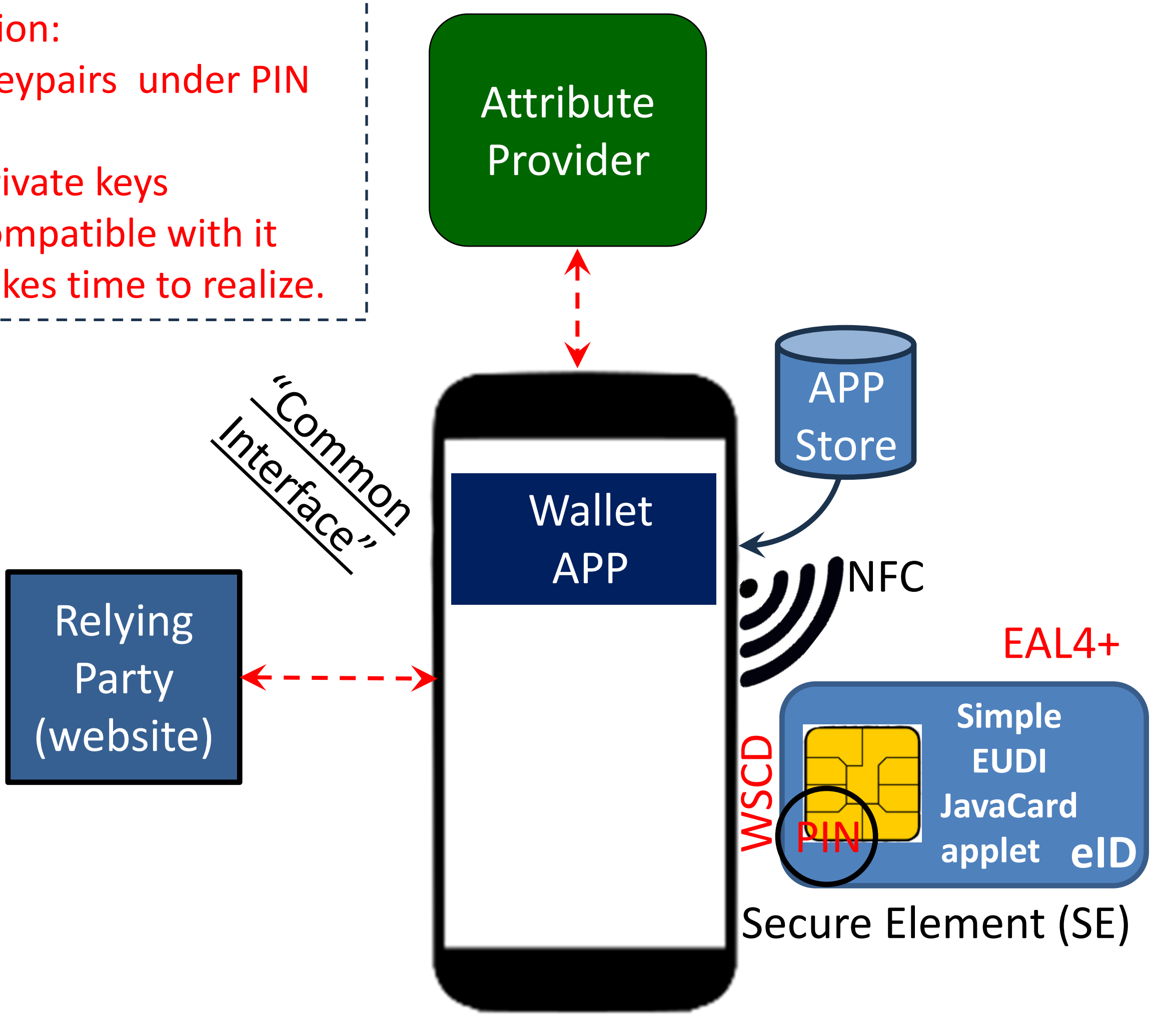
Outline

- EUDI-wallet based on eSIM and eID-smartcard (future end-state?)
- SECDSA properties
- The foundation: Split-ECDSA (SECDSA)
- SECDSA based HSM EUDI-wallet using standard mobile hardware
- Proof-of-Associations on standard mobile hardware and HSMs
- EUDI-wallet roadmap

EUDI-wallet based on eSIM and eID-smartcard (future end-state)

- Advanced JavaCard application:
  - Provides **all** attestation keypairs under PIN control
  - Delivers all attestation private keys
  - Uses no SECDSA but is compatible with it
- Technologically best, but takes time to realize.





See BSI_SAM_PositionPaper_v1-1.pdf

WSCD=Wallet Secure Cryptographic Device

eSE is always connected to Wallet APP → more difficult to secure than separate SE.

SECDSA properties

- Allows for EUDI-wallets based on native mobile cryptographic hardware (*) albeit probably not on eIDAS High assurance level by itself.
- Allows HSM assisted EUDI-wallet based on native mobile cryptographic hardware with properties:
  - eIDAS High assurance level (based on eIDAS1 notification process)
  - Optimal security (no information stored in wallet or stored/processed at WP allowing for PIN brute-force)
  - Support for publicly verifiable, non-reputable wallet instructions signatures 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 (*).

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

The foundation: Split-ECDSA (SECDSA)

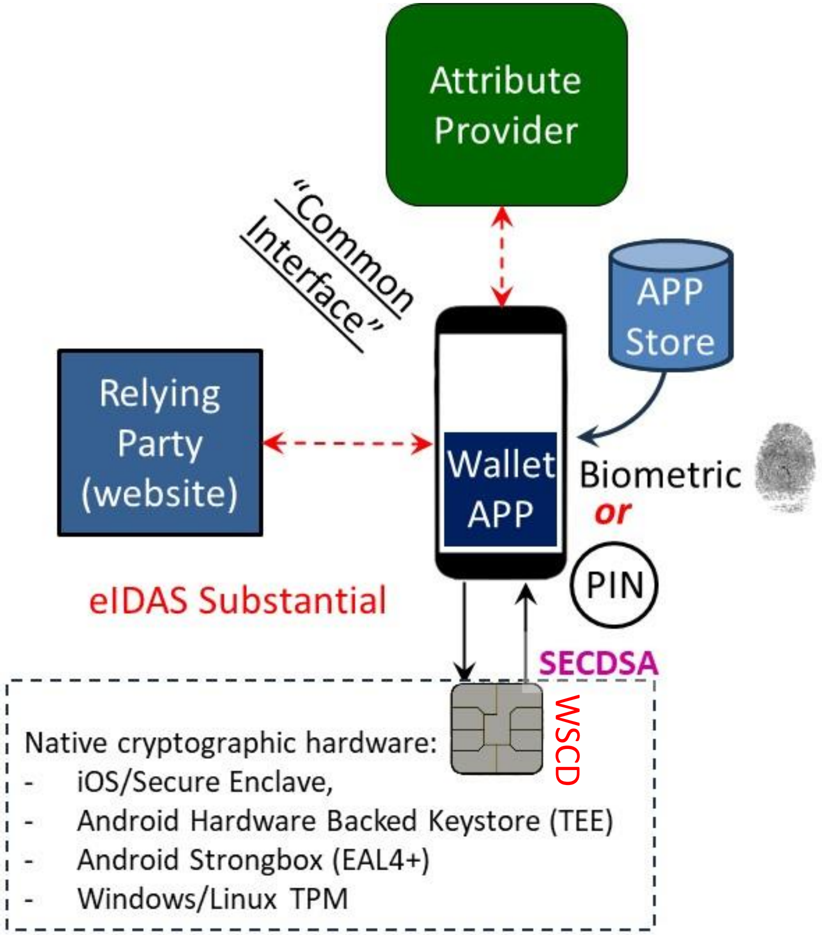
Algorithm 6 Split-ECDSA (SECDSA) signature generation
Input: message M, PIN-key $\sigma \in \mathbb{F}_q^*$, SCE-key $u \in \mathbb{F}_q^*$
Output: ECDSA signature (r, s) corresponding to public key $\sigma{\cdot}u{\cdot}G$.

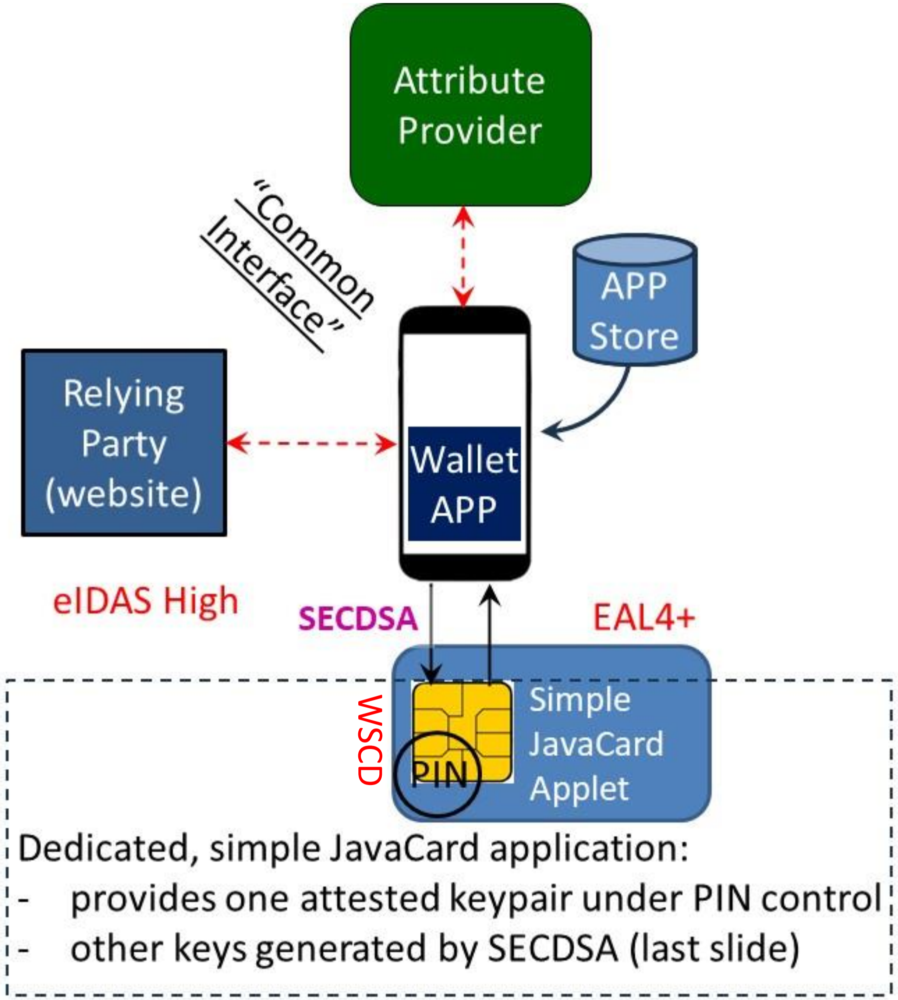
```
1: Compute H(M) and convert this to an integer e.
2: Compute e' = σ^{-1}·e mod q
3: Select random k ∈ {1, ..., q − 1}
4: Compute kG = (x, y) and convert x to integer x̄
5: Compute r = x̄ mod q. If r = 0 go to Line 1
6: If r mod q = 0 then go to Line 1
7: Compute s_0 = k^{-1}(e' + u·r) mod q. If s_0 = 0 go to Line 1
8: Compute s = σ·s_0 mod 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.
- SCE delivers an attested public key $\mathrm{U} = u \cdot G$ (with private key u).
- 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 $\mathbf{Y} = \sigma \cdot u \cdot G$ 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 $\sigma \cdot$ µ 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.
- Could be base for (next slide):
  - eIDAS substantial stand-alone EUDI-wallet based on native cryptographic hardware,
  - eIDAS High stand-alone EUDI-wallet based on simple smartcard application.

SECDSA-based stand-alone EUDI-wallet





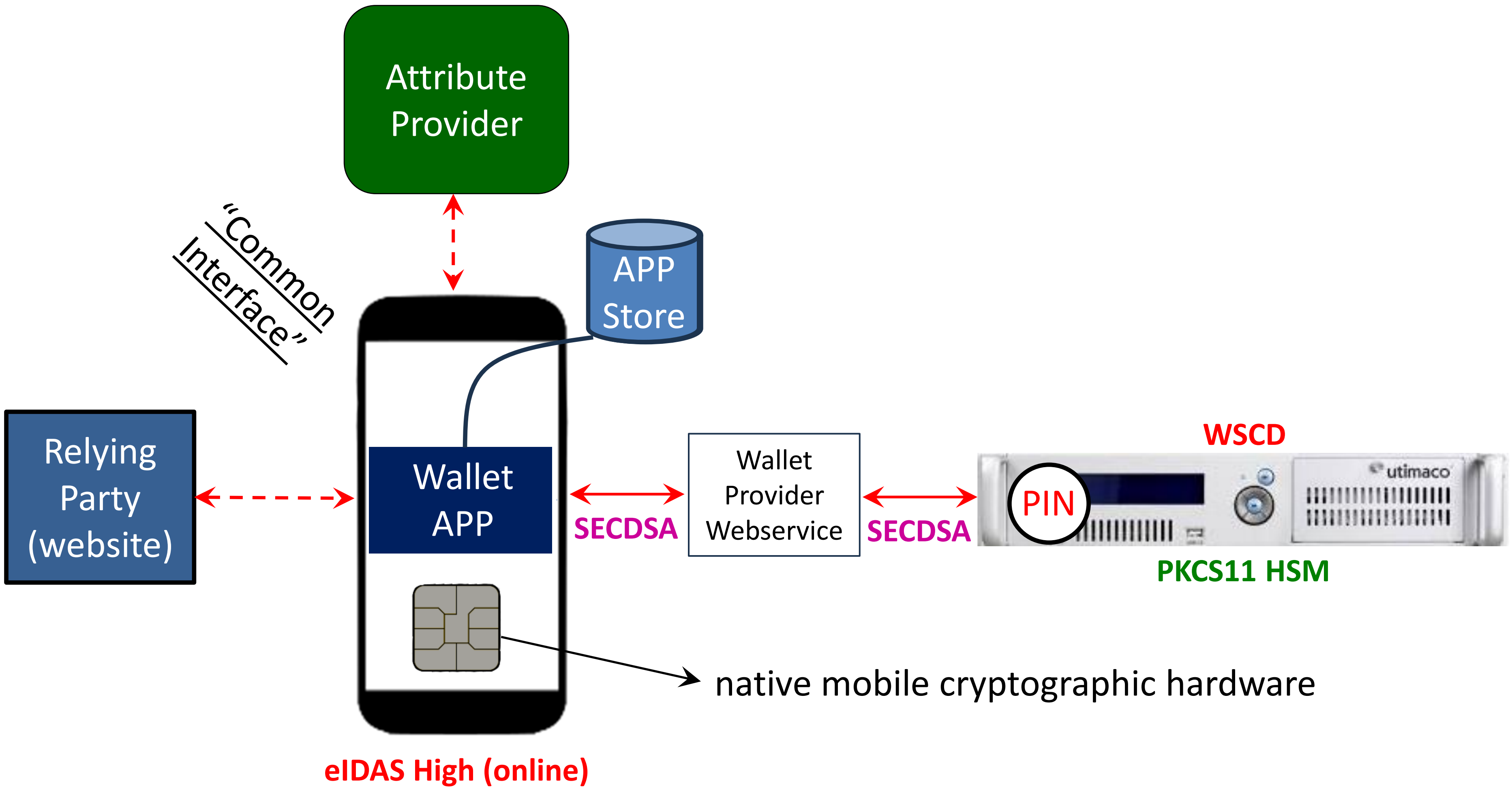
PIN: platform PIN or SECDSA PIN with brute-force protection (Section 3.4 SECDSA paper)

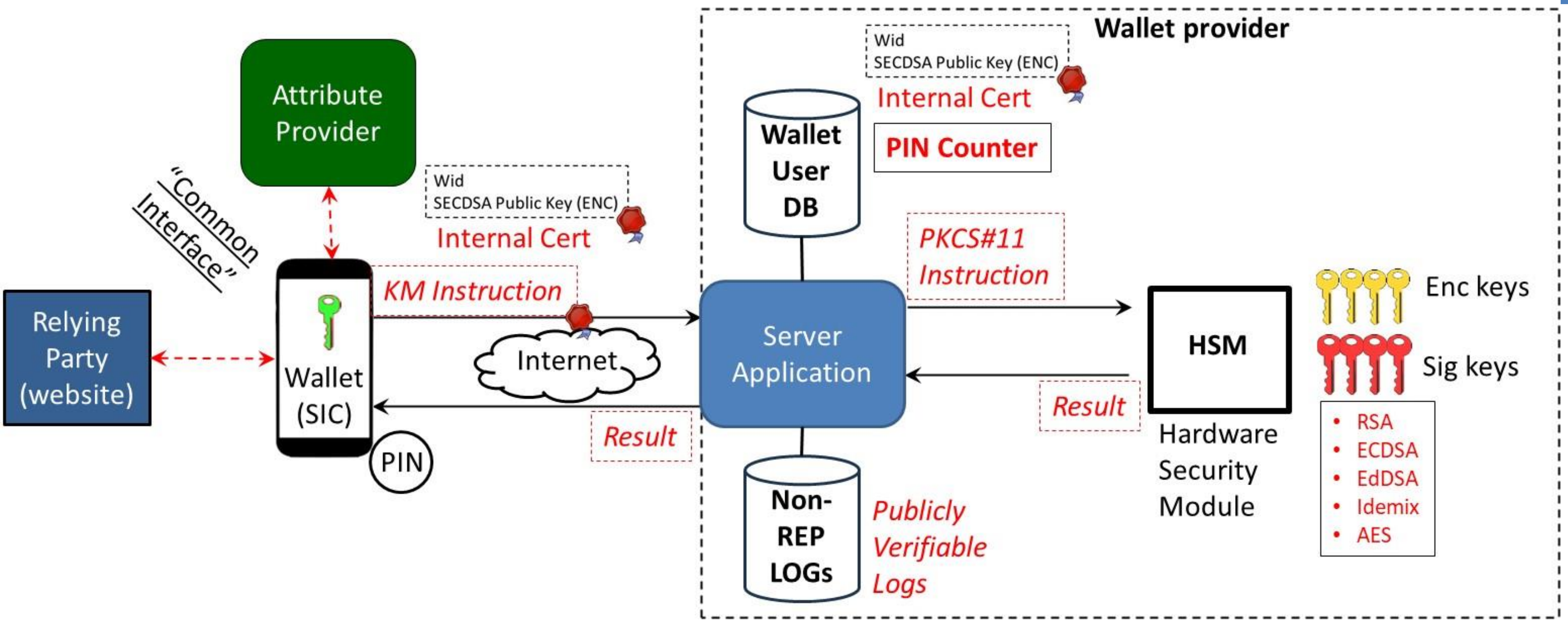
Biometric: σ-key under biometric control supported by platform (Section 3.5 SECDSA paper)

PIN: smartcard PIN

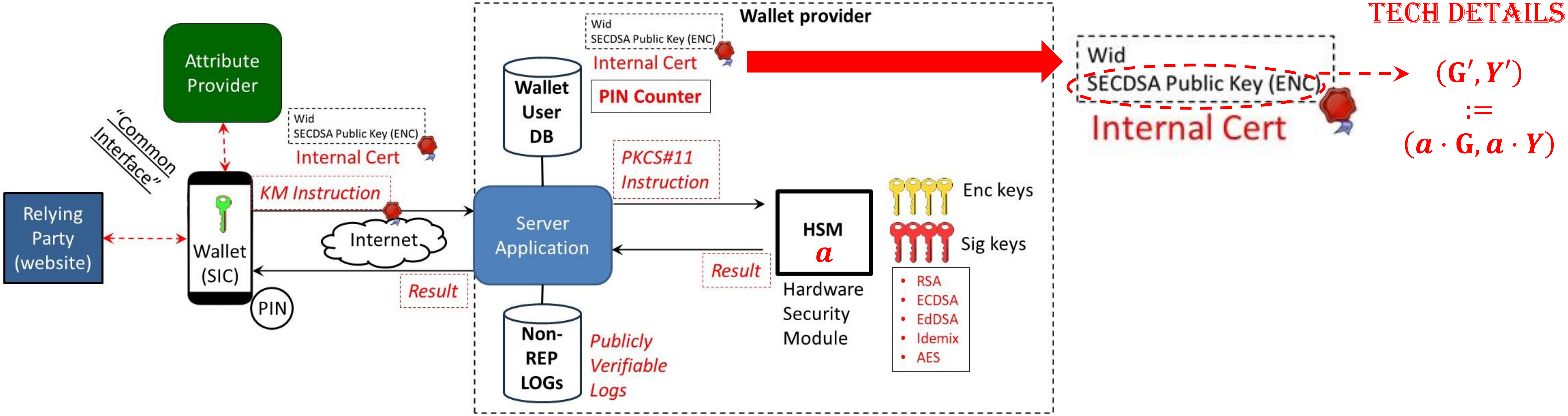
HSM assisted EUDI-wallet
based on native mobile cryptographic hardware

SECDSA based HSM EUDI-wallet using standard mobile hardware

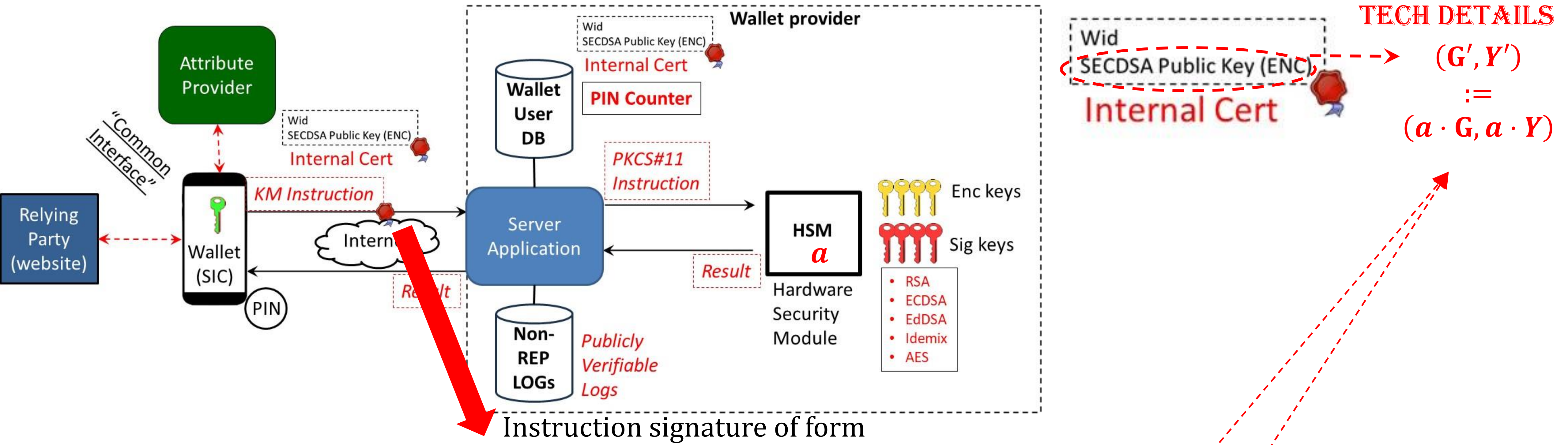




- During wallet initialisation, an *Internal Certificate* (IC) is agreed between wallet and wallet provider.
- Internal certificate holds unique Wallet Identifier (WId) and homomorphically encrypted raw SECDSA public key with DH key managed by the WP HSM to prevent PIN brute-forcing. Raw SECDSA Public key 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 $\boldsymbol{Y}$ takes form $(\boldsymbol{a} \cdot \mathbf{G}, \boldsymbol{a} \cdot \boldsymbol{Y})$ with secret scalar $\boldsymbol{a}$ 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 $\boldsymbol{a}$ (Diffie-Hellman key).



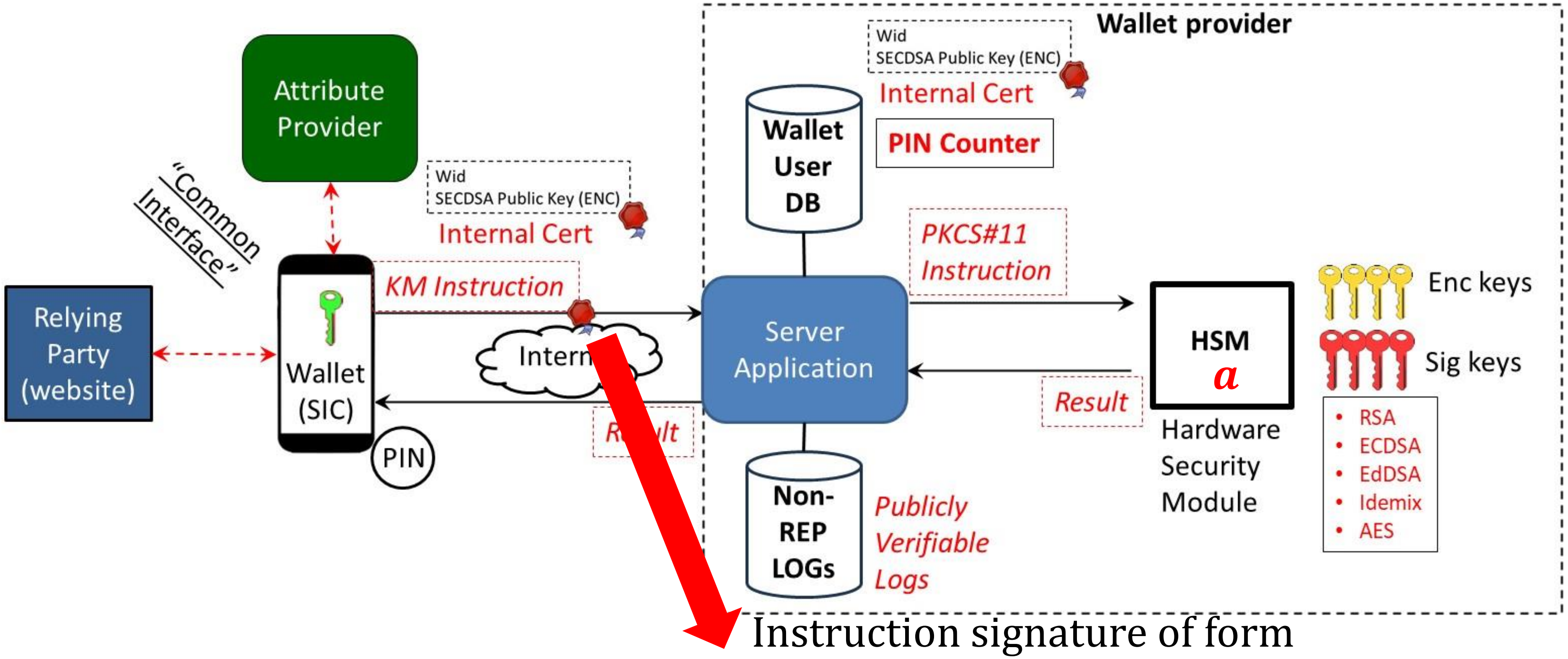
Instruction signature of form

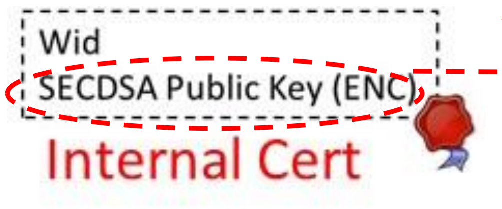
$(U,V,W) \coloneqq (R, s^{-1}\cdot G', s^{-1}\cdot Y')$ plus ZK1

The raw SECDSA signature (r,s) 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 ($\exists\, x: (V,W) = (x\cdot G', x\cdot Y')$).

TECH DETAILS





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

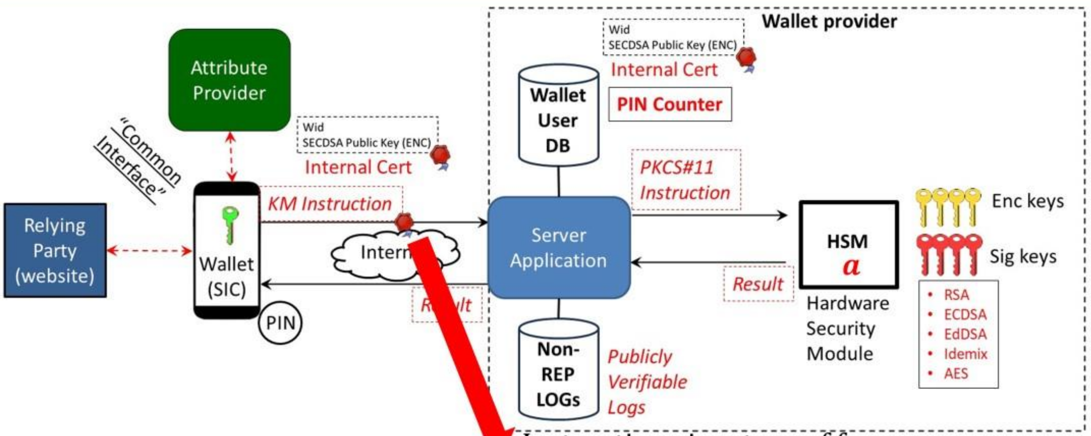
Instruction signature of form

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

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

1. The original r is reformed from U (i.e. 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



TECH DETAILS

Wid
SECDSA Public Key (ENC)
Internal Cert

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

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 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.

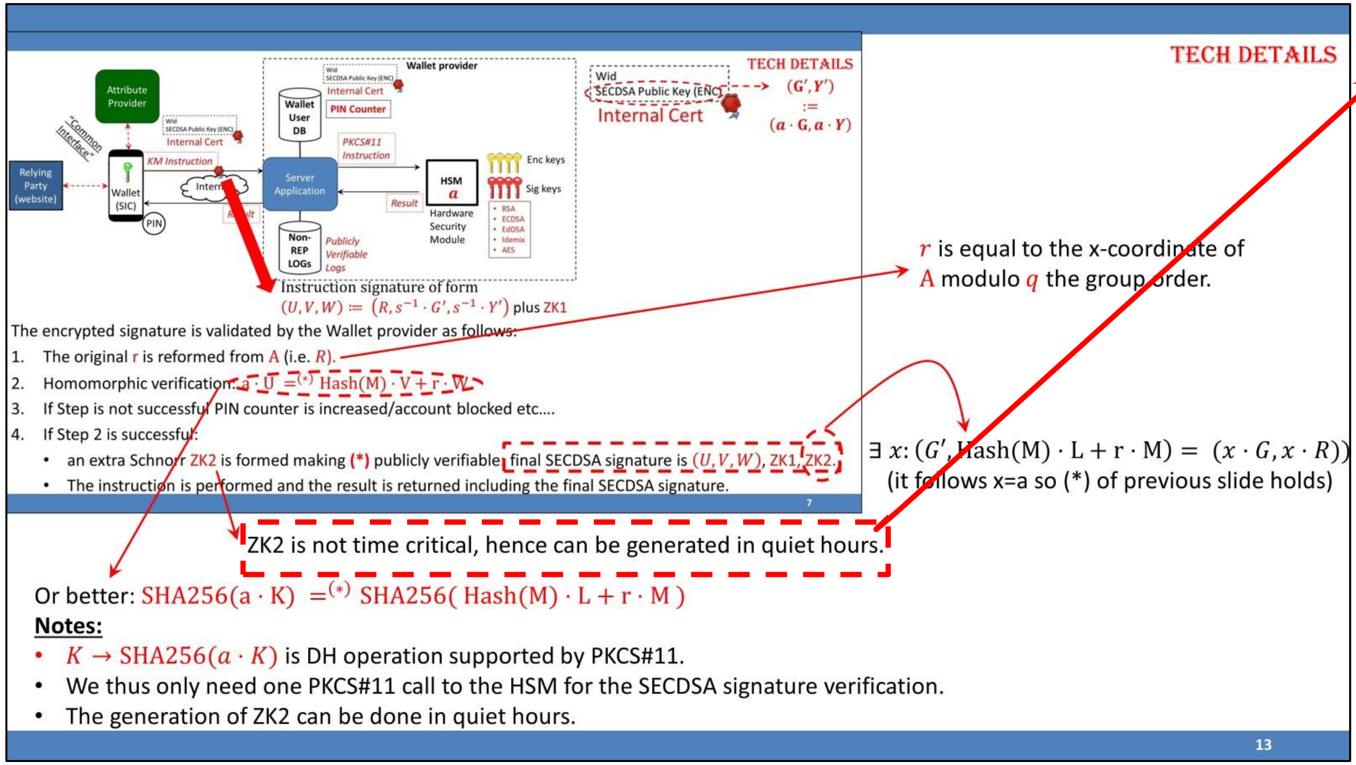
$\exists\, x: (G', \mathrm{Hash(M)} \cdot \mathrm{L} + \mathrm{r} \cdot \mathrm{M}) = (x \cdot G, x \cdot R))$
(it follows x=a so (*) of previous slide holds)

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

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

Notes:

- $K \to \mathrm{SHA256}(a \cdot K)$ 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.



- Schnorr Zero Knowledge Proof ZK2 is not PKCS#11 supported and requires a specific (but simple) HSM firmware module.
- Module has access to secret *a* (or the master key it if derived from the Wid which is better).
- Module input:
  WId, $(\mathrm{U}, \mathrm{V}, \mathrm{W}) = (G', R, \mathrm{Hash}(\mathrm{M}) \cdot \mathrm{L} + \mathrm{r} \cdot \mathrm{M})$
- Module looks up or derives secret *a* and checks if
  1. $U = a \cdot G$ (G is curve basepoint) and
  2. $W = a \cdot V$ both hold.

  If so, then the Modules generates the Schnorr Zero Knowledge Proof ZK2 to make this publicly verifiable and returns this.
- *Note that this Module does not allow an attacker to multiply random points with the secret a! That is, the Module is not a Diffie-Hellman Oracle.*

Android Studio project available.

09:32

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... → G’
- Encrypted_Pub_Key-Y part: 03e51f4ce244422... → Y’
- Certificate Signature: 68dd21935500e38...

(G’, Y’: Internal Certificate)

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... → R
- Encrypted s-part1: 03b8e2f3b2b059a... → $s^{-1} \cdot G'$
- Encrypted s-part2: 0246ff21d4f5408... → $s^{-1} \cdot Y'$
- Schnorr SF PoK r: 282af3442c2c120...
- Schnorr SF PoK s: 560b7f1e399ae9b...

(Schnorr SF PoK r, s → ZK1)

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...

(Schnorr RP PoK r, s → ZK2)

Qualified signature provided by User/WP to RP correct? true → Result of instruction
SECDSA evidence signature provided by User/WP to RP correct? true

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

Proof-of-Associations on standard mobile hardware and HSMs

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 (cryptographic hardware) and SCE is 'eIDAS' compliant.

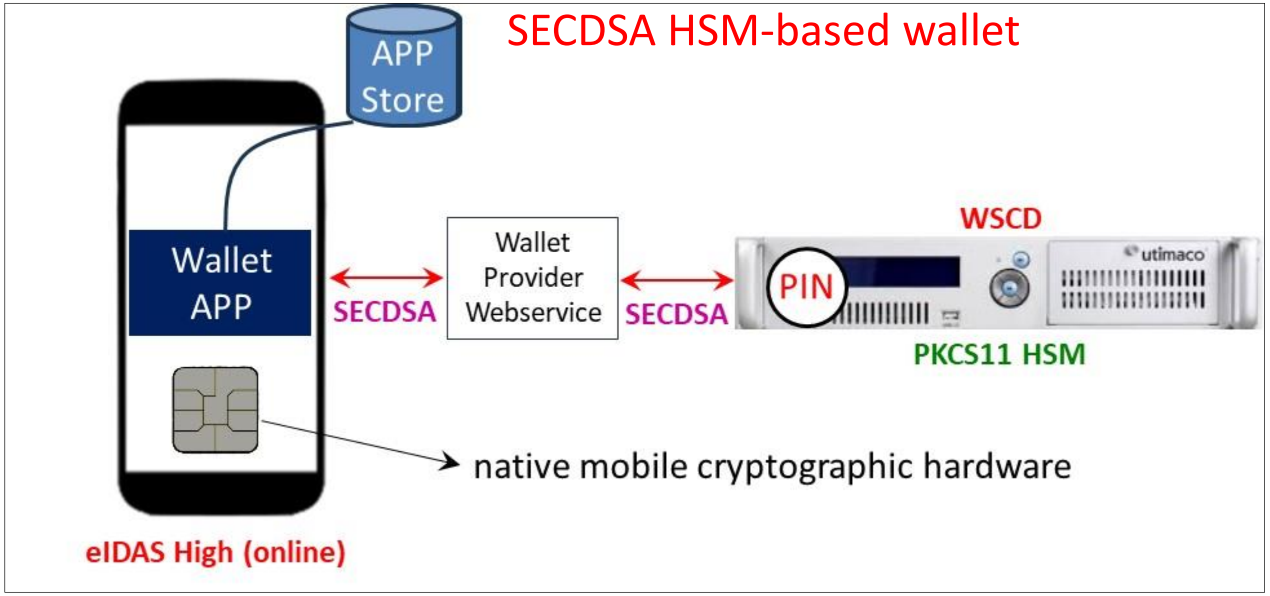
   Note: a WTA is typically issued by the Wallet Provider based on mobile platform (key) attestation capabilities.
3. The wallet/user can generate a public key V *associated* with the WTA public key U by generating a random scalar z and letting $\mathrm{V} = z \cdot U$. The scalar z could be static, derived 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 $v = z \cdot u$.
4. Wallet/user can prove two public keys U_1, U_2 are associated by proving possession of a private key y: $\mathrm{y} \cdot U_1 = U_2$.

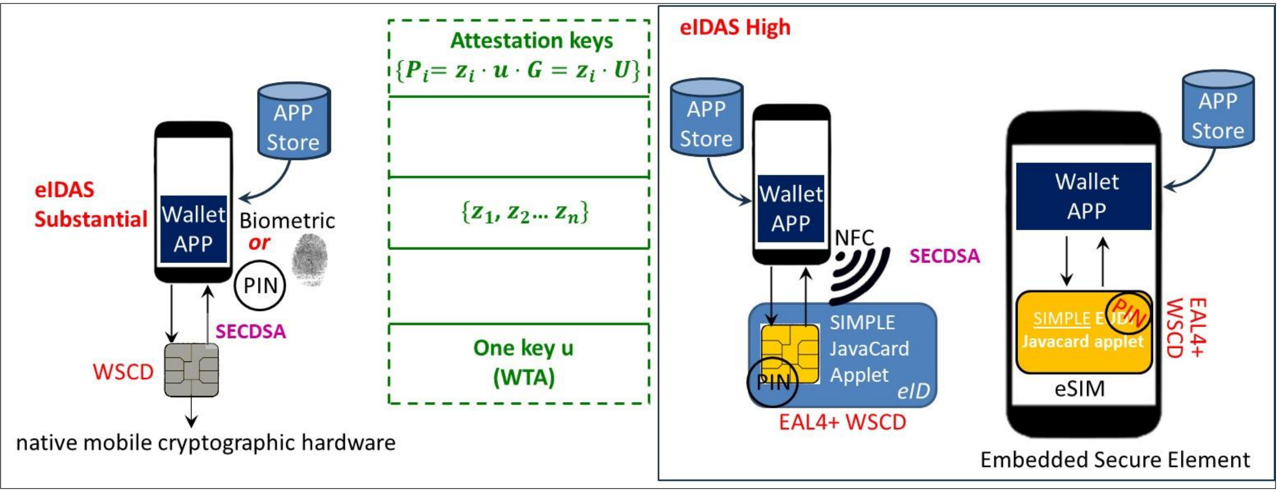
   Notes:
   - *If the public keys are $U_1 = k_1 \cdot U$, $U_2 = k_2 \cdot U$ are associated then $y = k_2\, k_1^{-1}$.*
   - *This proof can be given for instance using a Schnorr Zero-Knowledge Proof ('signature').*
   - *Associations are performed by (PID) issuers against the WTA public key, or against another public key that is known to be associated to this WTA key, e.g. a 'WTA copy'.*
   - *The proof of association of public keys always needs to be accompanied by a proof of possession of the keys involved; efficient combination is possible.*
5. Techniques are applicable to stand-alone wallets (standard mobile hardware) and HSM-based wallets (PKCS#11).

EUDI-WALLET ROADMAP

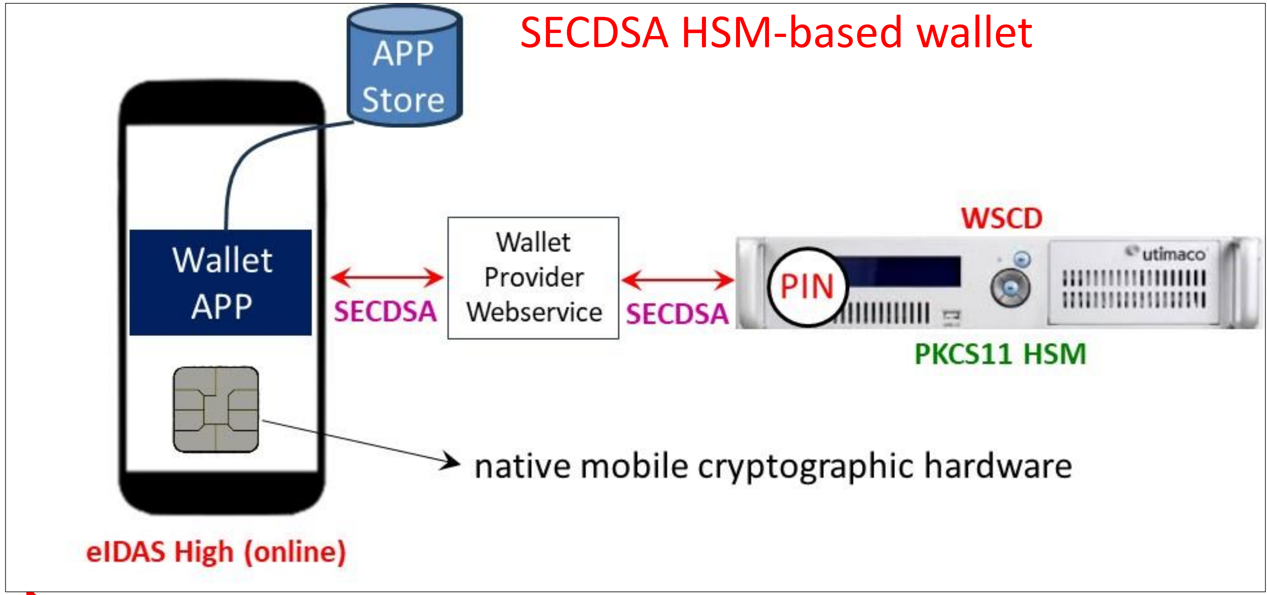
2026



SECDSA stand-alone wallet



> 2030



(e)SE based wallet