Authentication and sole control at a high level of assurance on widespread smartphones with threshold signatures

Sander Q. Dijkhuis
Cleverbase ID B.V.
sander.dijkhuis@cleverbase.com

February 18, 2025

Abstract

How to be assured that a user entered their PIN on their smartphone? The question is especially relevant when deploying remotely secured services such as with mobile wallets for digital identity and banking, which typically deploy a server side backed by a hardware security module (HSM). As long as the server can be trusted, authentication can be performed with high assurance, but it is challenging to guarantee sole control. This report defines an approach in terms of an abstract security problem and a concrete solution based on threshold signatures. It can be applied to use cases such as HSM-backed mobile identity wallets and other identification means.

1 Introduction

Increasingly, social and economic interactions occur online, increasing the potential impact of cybersecurity threats including threats to privacy. This requires enhanced protection of data against unauthorised access and disinformation. Recent public and private policies therefore require a high assurance level with regard to the user authentication and control, and take measures to enable high adoption of secure cryptographic devices among the potential user base.

For example, the European Digital Identity Regulation [12] requires Member States to provide all natural and legal persons access to authentic identity attributes. To ensure large-scale adoption, the regulator aims to leverage the widespread ownership of smartphones. On these, users install an app providing a European Digital Identity Wallet, which provides authentication at a high assurance level according to common baseline specifications [3], as well as sole control over privileged operations.

A challenge to this adoption is that many common smartphones do not yet contain sufficiently secure hardware to meet such requirements, and even when they do, the license to use this hardware may be unavailable. To accelerate adoption, the app may therefore rely on security functions provided via a secure

channel by remote hardware, such as a centrally managed hardware security module (HSM). Such an approach is for example foreseen in the Dutch ministerial order on identification means [11], and is common in online banking to meet Strong Customer Authentication requirements [5]. A well-known approach to solve the authentication problem is based on public key cryptography: the HSM protects the user's private key within its tamper-protected environment, and only activates the private key upon verification of the secure channel and rate-limited entry of the user's PIN. A verifier authenticates the user with signatures created this way against the user's enrolled public key.

However, the remotely secured approach can pose new threats to sole control: the user is not able to monitor the remote server and HSM and detect malicious modifications to its software. For example, the remote service provider may maliciously or unknowingly install a backdoor to enable activation of the user's private key without requiring entry of the user's PIN, taking over access to their protected data. In the example context of government-regulated identity wallets, such backdoors may be abused for law enforcement or identity fraud, in both cases disrupting the policy objectives.

This technical report presents an approach to achieve both high assurance level authentication and sole control using common smartphones backed by remote servers. Key security features are multi-factor authentication, context binding, non-repudiation and transparency: each authentication process results in publicly verifiable evidence that the smartphone was used with the user's PIN. The evidence is non-repudiable and potentially bound to application context for sole control, such as an instruction or an authorization scope.

The first section models the cryptographic capabilities of common smartphones and servers which could provide assurance. The second section defines the approach in abstract terms of a security problem, enabling multiple solutions. The third section presents a solution to this security problem applying threshold signatures. The fourth section presents example applications of solutions.

2 Related work

A well-known building block for authentication systems is a threshold signature scheme, such as Flexible Round-Optimized Schnorr Threshold Signatures (FROST) [9]. However, this is not sufficient for widespread smartphones, since their hardware does not natively support threshold signing.

A first system design meeting similar goals as in this report is based on split key ECDSA signatures [18]. However, this approach requires verifiers to implement low-level group operations to verify homomorphically encrypted verification data. Without the homomorphic encryption, an adversary with access to the smartphone could use the credentials to verify guesses at the user's PIN.

To open up the design space for alternative optimisations, this report contributes an analysis of the security problem and proposes an alternative solution that optimises for simplicy on the verifier side.

3 Prerequisites

Common smartphones provide two relevant execution environments: the programmable application environment, and the static secure area. Examples of the secure area are on Android-based phones the Trusted Execution Environment (TEE) and StrongBox, and on iPhone models the Secure Enclave. Applications can request services from the secure area using common interfaces. Both environments can perform cryptography, but only the secure area is trusted to provide non-extractable private or secret keys.

In the context of HSMs, a distinction is made between several execution environments as well. For simplicity, this report also assumes a static secure area and a programmable application environment for the local or external client application.

Regulation typically limits what capabilities can be evaluated for cybersecurity. For example, EU Member States rely on SOG-IS agreed cryptographic mechanisms [15] for cross-border assurance.

This section models the relevant cryptographic capabilities with assurance.

Cryptographic hash Application environments can implement cryptographic hash functions using appropriate libraries, providing an algorithm:

- $\#(\mathsf{msg})$: outputs a pseudo-random byte string of fixed size.

In the context of smartphones, the most common hash algorithm is SHA-256 [16].

Message authentication Some secure areas implement a keyed-hash message authentication code (HMAC) [17], which includes the algorithms:

- $\mathsf{SecretGen}()$: outputs a randomized HMAC secret key k.
- $\mathsf{MAC}(k, \mathsf{msg})$: deterministically outputs a fixed-size MAC using secret key k on message msg.

Prime-order group All common secure areas at least a common group $\mathbb{G}$ of prime order q with base point $G \in \mathbb{G}$, such as the P-256 elliptic curve [13]. This report uses additive notation and denotes the scalar field as $\mathbb{F}_q^*$. Implementations include at least the following algorithms:

- $\mathsf{KeyGen}()$: outputs a randomized key pair $(d, [d]G) \in (\mathbb{F}_q^*, \mathbb{G})$.

Application environments can additionally support additional group and scalar operations using a cryptographic library. These include the following algorithms [6]:

- $\#_{\mathsf{C}}(\mathsf{msg})$: outputs a cryptographic hash in $\mathbb{G}$ of byte string msg.
- $\#_{\mathsf{F}}(\mathsf{msg})$: outputs a cryptographic hash in $\mathbb{F}_q^*$ of byte string msg.

Key agreement The Secure Enclave and some Android secure areas provide an implementation of Elliptic Curve Key Agreement with the Diffie-Hellman protocol (ECKA-DH) [14] on $\mathbb{G}$. This includes the following algorithm:

- $\mathsf{DH}(d, P)$: outputs the byte string representation of the x coordinate of $[d]P$ for scalar $d \in \mathbb{F}_q^*$ and element $P \in \mathbb{G}$. The representation is big-endian and fixed-size.

Digital signatures All secure areas implement a digital signature algorithm on $\mathbb{G}$, which can be publicly verified using the following algorithms:

- $\mathsf{Verify}(D, \mathsf{msg}, \mathsf{sig})$: outputs 1 if sig is a signature over msg for public key D, 0 otherwise.

The secure area includes, at least for the Elliptic Curve Digital Signature Algorithm (ECDSA) [14] on P-256 with SHA-256 hashing:

- $\mathsf{Sign}(d, \mathsf{msg})$: outputs a signature sig such that $\mathsf{Verify}([d]G, \mathsf{msg}, \mathsf{sig})$ returns 1.

In ECDSA, a signature is modeled as $\mathsf{sig} = (r, s) \in \mathbb{F}_q^* \times \mathbb{F}_q^*$. Application environments can additionally support Schnorr signatures using a cryptographic library. One variant is the Elliptic Curve-based Schnorr Digital Signature Algorithm (ECSDSA) [14] over P-256 with SHA-256, which models signatures as $\mathsf{sig} = (c, s)$ where c is a fixed-size byte string and $s \in \mathbb{F}_q^*$. Another variant is the Edwards-Curve Digital Signature Algorithm (EdDSA) [8] which models signatures as $\mathsf{sig} = (R, s) \in \mathbb{G} \times \mathbb{F}_q^*$.

4 Security problem

A remotely protected smartphone authenticator for high-assurance authentication and sole control is modelled as two components with rate-limited communication:

- client: a smartphone app with user interaction and limited access to the smartphone secure area;
- server: a rate-limited server application.

Relying on the available cryptographic prerequisites, the problem is to protect the following primary assets:

- identification data: data that identifies the user, consisting of:
  - possession factor: smartphone private key $d_{\mathsf{p}} \in \mathbb{F}_q^*$ in a digital signature algorithm
  - knowledge or inherence factor: high-entropy activation data modelled as private key $d_{\mathsf{a}} \in \mathbb{F}_q^*$
- context data: byte string ctx representing an application context, for example including a login challenge or an access scope to authorize

The primary objective is to enable any third-party verifier to verify the identification data in a way that is cryptographically bound to the context data, relying on secondary assets:

- verification data: to be designed data that enables verification of the identification data, consisting of:
  - credential: a static tuple cred consisting of one or more public keys, to be designed;
  - transcript: a dynamically system-generated tuple tr proving the possession and second factor, bound to the context data, to be designed;
- protection data: high-entropy user-specific private key $d_\mathsf{s} \in \mathbb{F}_q^*$ held at the server, for example derived using $d_\mathsf{S} \leftarrow \#_\mathsf{F}(\mathsf{HMAC}(k, D_\mathsf{P}))$ using an HSM-protected secret key k;
- protocol data: to be designed data exchanged between client and server.

Only the user interacts directly with the system, physically protecting d_p and willingly providing d_a to the smartphone app while being informed about ctx. For example:

- the user enters a PIN from which the app derives
  - $d_\mathsf{a} \leftarrow \#_\mathsf{F}(\mathsf{HMAC}(k, \mathsf{PIN}))$ using a secure area secret key k; or
  - $d_\mathsf{a} \leftarrow \#_\mathsf{F}(\mathsf{DH}(d, \#_\mathsf{C}(\mathsf{PIN})))$ using a secure area private key d; or
- the smartphone's secure area evaluates live-recorded biometric data to enable decryption of d_a from a local key store.

The system aims to protect against these main threats:

- information disclosure, affecting identification data and protection data confidentiality;
- transcript forgery, affecting context data, protocol data and verification data integrity.

The model assumes that:

- d_p is well-protected by the smartphone hardware;
- d_a is well-protected by the user and the rate-limiting mechanism;
- prot is well-protected by the server provider's security controls, for example relying on a hardware security module (HSM);
- cred is well-protected against forgery and includes the appropriate public keys;
- ctx includes replay attack prevention data if needed, such as a nonce or a timestamp.

The system has the following security objectives. To meet a high authentication assurance level, objectives are derived from the common baseline specifications [3] on the electronic identification means characteristics and design (Annex, Section 2.2.1) and on the authentication mechanism (Annex, Section 2.3.1). To meet a high sole control assurance level (SCAL), objectives are derived from the European Standard [1] on trustworthy systems supporting server signing. While the standard defines two levels SCAL1 and SCAL2 based on a substantial authentication assurance level, the set of requirements in this section can be considered a higher level SCAL3 [2].

Multi-factor The verification transcript is highly likely to be created during a single process during which all identification data were available. That is, given oracles that output d_{p} signatures, protocol data and transcripts, an efficient adversary is highly unlikely to forge new protocol data and verification data; and given d_{a}, protocol data and transcripts, an efficient adversary is highly unlikely to new protocol data or verification data.

Context-bound The context data is highly unlikely to be modified after authentication. That is, given an oracle that outputs transcripts, an efficient adversary is highly unlikely to forge new context data that can be verified successfully.

Non-repudiable It is highly unlikely that anyone, including the server, has forged verification data without control over all identification data components. That is, given oracles that output protocol data and transcripts, an efficient adversary is highly unlikely to forge new verification data.

Transparent The verification data can safely be provided to anyone for verification, not just to the server, without compromising protection against the main threats. That is, given access to a credential and to the smartphone app, an efficient adversary is highly unlikely to guess the PIN.

5 Applying threshold signatures*

This report presents the following solution to the security problem. It applies FROST [9] [4] as an extension to digital signature algorithm $\Sigma_{\mathsf{A}} = (\mathsf{Sign}, \mathsf{Verify})$, including the following algorithms:

- $\mathsf{Commit}()$: probabilistically outputs $((d, e), (D, E))$, a pair of nonces $d, e \in \mathbb{F}_q^*$ and commitment shares $([d]G, [e]G)$.
- $\mathsf{AggregateFirst}(\mathsf{msg}, P, C)$: deterministically outputs the first part of the group signature on message msg verifiable with group public key $P \in \mathbb{G}$ based on commitment list C, which is a tuple of tuples (i, D_i, E_i) with participant index i and commitment shares $D_i, E_i \in \mathbb{G}$.

*Patent NL2037022 pending.

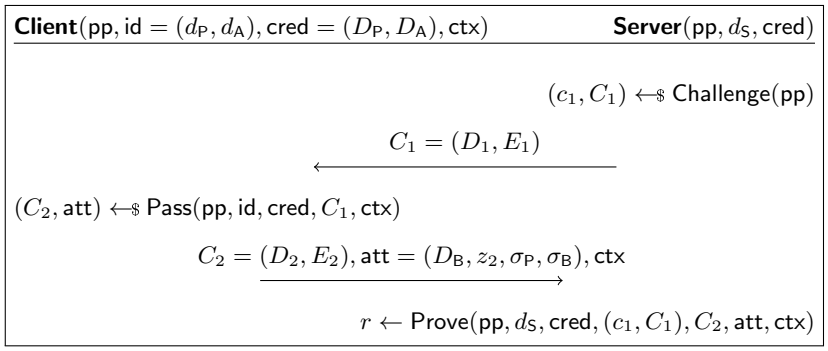

Figure 1: The authentication protocol applying threshold signatures, resulting in activation signature $r = \sigma_{\mathsf{A}}$ or rejection $r = \perp$. Each challenge tuple C_1 can be used only once, and access to the Prove endpoint is rate-limited after verification of possession signature σ_{P} using cred. Integrity of each attempt att is protected using binding signature σ_{B}, to be verified using D_{B}.

- $\mathsf{SignShare}(\mathsf{msg}, s, (d, e), C)$: deterministically outputs z, a signature share on message msg using signing key share s, nonces $d, e \in \mathbb{F}_q^*$ and commitment list C, which is a tuple of tuples (i, D_i, E_i) with participant index i and commitment shares $D_i, E_i \in \mathbb{G}$.
- $\mathsf{Aggregate}(\mathsf{msg}, C, Z)$: deterministically outputs a "FROSTy" Schnorr signature on message msg using a commitment list C and a signature share list Z, which is a tuple of tuples (i, z_i) with participant index i and signature share $z_i \in \mathbb{F}_q^*$.

After initial setup, the solution consists of two protocol steps between client and server, after which any verifier can check the authentication. The protocol steps are illustrated in Figure 1 and specified along with the setup and check algorithms in Figure 2. In this solution, the credential consists of the public keys $\mathsf{cred} = (D_{\mathsf{P}}, D_{\mathsf{A}})$ and the transcript of $\mathsf{tr} = (D_{\mathsf{B}}, \mathsf{dgst}, \sigma_{\mathsf{P}}, \sigma_{\mathsf{B}}, \sigma_{\mathsf{A}})$. A prototype is available at [2].

The remainder of this section presents arguments demonstrating that the solution meets the objectives.

Proposition. *The solution meets the Multi-factor objective.*

Proof. Consider the scenarios of individual authentication factor compromise.

If only d_{P} access is compromised, the adversary is highly unlikely to forge transcripts since these would require either a new FROSTy signature σ_{A}^* over fresh D_{B}^*, or knowledge of the discrete logarithm of a previous D_{B}, both of which are highly unlikely.

If only d_{A} access is compromised, the adversary is also highly unlikely to forge transcripts since these would involve fresh server nonces and commitments, so $\sigma_{\mathsf{A},1}$ would be different, and the adversary is highly unlikely to forge the proof of possession σ_{P}^* over this data.

If access to both d_{A} and d_{S} is compromised, the adversary could only efficiently succeed by replaying a previous σ_{P}, and is thereby bound to $\sigma_{\mathsf{A},1}$. But

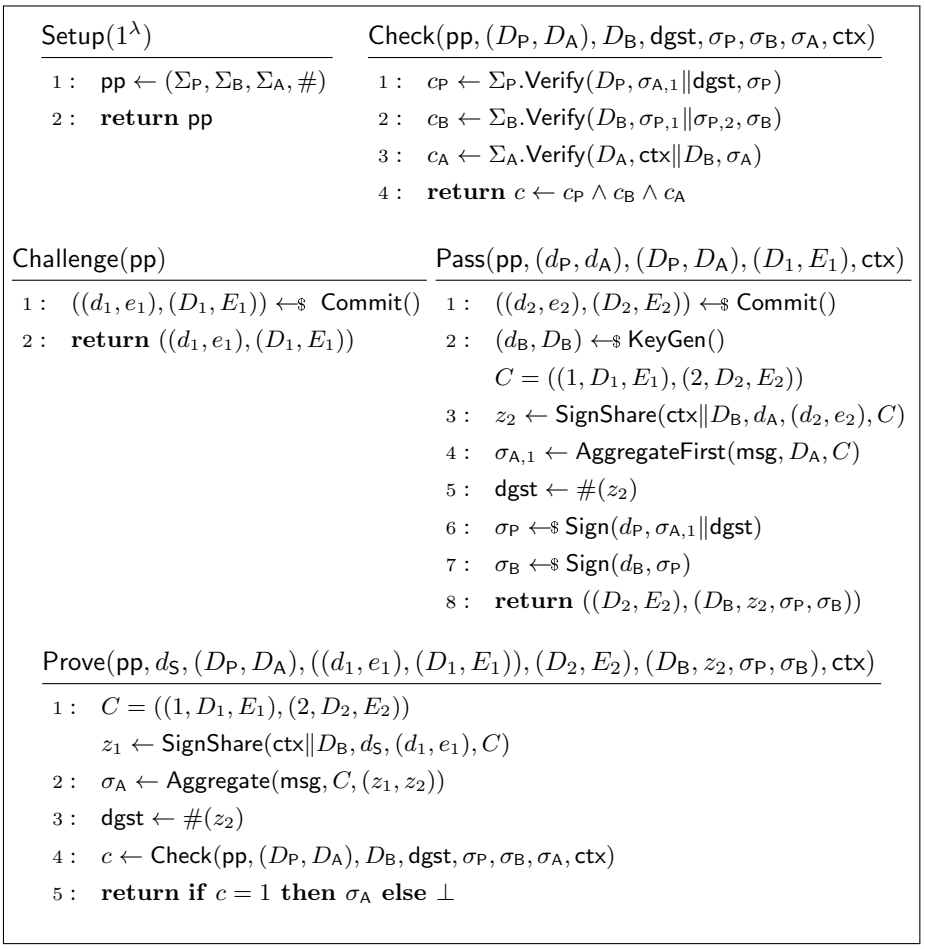

Figure 2: The algorithms for the patent-pending authentication method applying threshold signatures.

this first component of the FROSTy signature is computed using the cryptographic hash of the $\mathsf{ctx}\|D_B$, and it is highly unlikely the adversary finds a second preimage of this hash. So the adversary could only use d_A and d_S to arrive at the same signature σ_A, and not a successful forgery. □

Proposition. *The solution meets the Context-bound objective.*

Proof. Integrity of the context data ctx is protected by σ_A and subsequently by σ_P. □

Proposition. *The solution meets the Non-repudiable objective.*

Proof. By security of FROSTy signatures, even an efficient adversary with access to d_S could not forge σ_A^*. Therefore the adversary is bound to D_B, of which it is infeasible to find the discrete logarithm, and therefore the adversary is bound to σ_B and therefore σ_P. □

Proposition. *The solution meets the Transparent objective.*

Proof. Even with access to D_{A} and the method to derive a candidate d'_{A}, for example using the smartphone's message authentication or key agreement key, an adversary would need information about d_{S} to verify if they have found the discrete logarithm $d_{\mathsf{A}} + d_{\mathsf{S}}$ of D_{A}. But the server only returns zero-knowledge proofs, so it is unlikely that a single guess provides sufficient information to inform a second guess. In practice, the rate-limiting also limits the amount of PIN attempts, therefore minimising the risk. □

6 Example applications

The remotely protected smartphone authenticator can be applied in systems where the verifier is the server, and performs privileged operations upon authorization. Examples are a qualified signature creation device [1] or a wallet secure cryptographic application, both of which protect the user's private keys.

An alternative application is distributed: the verification data is verified as evidence by a third device. For example, the authentication server may be part of a high-assurance authorization server such as in OAuth [7], recording the evidence for auditing. Either the associated resource servers could perform such auditing automatically, or it can be performed in a separate process by a separate entity.

In another application, the verification data is recorded in transparency logs, such as in Certificate Transparency [10]. This enables users to monitor whether all verification data are recorded, and claim damage when finding evidence of a privileged operation without finding recorded verification data that authorizes that operation. Instead of a public record, the log may be protected or only be shared in the case of actual disputes. This principle is further elaborated in [18].

7 Conclusion

This report proposes a formalisation of the security problem involved with applying widespread smartphones for assurance level high authentication and sole control. It demonstrates that this problem can be solved at least by applying threshold signatures. This contributes a solution that optimises for simplicy on the verifier side.

References

[1] *CEN EN 419 241-1: Trustworthy Systems Supporting Server Signing - Part 1: General System Security Requirements*. 2018.

[2] Cleverbase. *Verify that systems operate under your sole control*. 2024. URL: `https://github.com/cleverbase/scal3`.

[3] *Commission Implementing Regulation (EU) 2015/1502 of 8 September 2015 on setting out minimum technical specifications and procedures for assurance levels for electronic identification means pursuant to Article 8(3) of Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market*. URL: https://data.europa.eu/eli/reg_impl/2015/1502/oj.

[4] Deirdre Connolly et al. *The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures*. RFC 9591. June 2024. DOI: 10.17487/RFC9591. URL: https://www.rfc-editor.org/info/rfc9591.

[5] *Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market, amending Directives 2002/65/EC, 2009/110/EC and 2013/36/EU and Regulation (EU) No 1093/2010, and repealing Directive 2007/64/EC*. URL: https://data.europa.eu/eli/dir/2015/2366/oj.

[6] Armando Faz-Hernandez et al. *Hashing to Elliptic Curves*. RFC 9380. Aug. 2023. DOI: 10.17487/RFC9380. URL: https://www.rfc-editor.org/info/rfc9380.

[7] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012. DOI: 10.17487/RFC6749. URL: https://www.rfc-editor.org/info/rfc6749.

[8] Simon Josefsson and Ilari Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. Jan. 2017. DOI: 10.17487/RFC8032. URL: https://www.rfc-editor.org/info/rfc8032.

[9] Chelsea Komlo and Ian Goldberg. *FROST: Flexible Round-Optimized Schnorr Threshold Signatures*. Cryptology ePrint Archive, Paper 2020/852. 2020. URL: https://eprint.iacr.org/2020/852.

[10] Ben Laurie, Eran Messeri, and Rob Stradling. *Certificate Transparency Version 2.0*. RFC 9162. Dec. 2021. DOI: 10.17487/RFC9162. URL: https://www.rfc-editor.org/info/rfc9162.

[11] *Regeling nadere eisen identificatiemiddelen, authenticatiediensten en machtigingsdiensten Wdo*. Dutch. URL: https://zoek.officielebekendmakingen.nl/stcrt-2025-4198.

[12] *Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework*. URL: https://data.europa.eu/eli/reg/2024/1183/oj.

[13] Certicom Research. *SEC 2: Recommended Elliptic Curve Domain Parameters*. 2010. URL: https://www.secg.org/sec2-v2.pdf. Version 2.0.

[14] Bundesamt für Sicherheit in der Informationstechnik (BSI). *TR-03111: Elliptic Curve Cryptography*. 2018. URL: https://www.bsi.bund.de/dok/TR-03111-en. Version 2.10.

[15] *SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms*. 2020. URL: http://sog-is.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf. Version 1.2.

[16] National Institute of Standards and Technology. *FIPS 180-4: Secure Hash Standard (SHS)*. 2015. URL: https://doi.org/10.6028/NIST.FIPS.180-4.

[17] National Institute of Standards and Technology. *FIPS 198-1: The Keyed-Hash Message Authentication Code (HMAC)*. 2008. URL: https://doi.org/10.6028/NIST.FIPS.198-1.

[18] Eric Verheul. *An HSM-based EUDI wallet using Split-ECDSA (SECDSA) providing verifiable "sole control"*. 2024. URL: https://wellet.nl/SECDSA-EUDI-walletV2.pdf. Version 13 October 2024.