Deniability for Signed Credentials: Revisiting the Authenticated Channel vs. Signed Data Debate in the EUDI Wallet *

Magdalena Bertram[1] and Anja Lehmann[2]

[1] Fraunhofer AISEC, Berlin, Germany magdalena.bertram@aisec.fraunhofer.de
[2] Hasso Plattner Institute, University of Potsdam, Germany anja.lehmann@hpi.de

Abstract. The European Digital Identity Wallet (EUDI Wallet) is currently adopting ECDSA-based signed credentials as part of its core architecture, which raised concerns that such designs inherently lack plausible deniability compared to authenticated-channel approaches such as the German electronic identity card. This paper revisits this perceived trade-off and argues that it is not a property of signature schemes themselves, but of the credential presentation protocol. We show that standard cryptographic techniques – specifically lightweight OR-proofs over the native ECDSA verification equation – can be used to transform signed credential presentations into non-transferable, verifier-bound transcripts. Our contribution is not a new cryptographic primitive, but a careful instantiation of well-established techniques within the EUDI context, showing that deniability can be added to signed credentials while preserving their deployment advantages.

1 Introduction

The European Digital Identity Wallet (EUDI-Wallet) is a major ongoing deployment of a digital identity infrastructure in the European Union. By the end of 2026, every EU Member State is required to provide at least one EUDI Wallet to its citizens for storing and presenting identity documents, attribute attestations, and other credentials [17]. The technical architecture is currently being finalised, with the Architecture and Reference Framework [15] serving as the de facto standard for implementations across Member States.

A central design decision concerns how the authenticity and integrity of presented identity data is guaranteed. Two fundamentally different approaches have been discussed (in Germany): *authenticated channels* and *signed credentials*. In the authenticated-channel approach – the mechanism underlying the German electronic identity card (*Personalausweis*) [7,6] – a trusted chip on the user's device runs an interactive protocol with the verifier that establishes a shared symmetric session key and authenticates the chip's possession of an issuer-certified

* This is an extended version of a paper accepted for publication in *Availability, Reliability and Security — ARES 2026 Workshops*, Lecture Notes in Computer Science, vol. 16900, Springer Nature. This version includes full security proofs as additional material.

key; the attributes are then transmitted over the resulting authenticated channel. In the signed-credentials approach, the issuer produces a publicly verifiable signature over the user's attributes that the wallet stores and forwards to a verifier at presentation time, with no issuer interaction needed at presentation. In October 2024, Germany settled this debate for its national EUDI-Wallet by deciding in favour of signed credentials, specifically ECDSA-signed SD-JWT credentials [8]. The decision drew criticism from consumer and data protections organisations [29,14,30,13], who argue that signed credentials lack a privacy property that the authenticated-channel design of the eID provides: *plausible deniability*, the property that a verifier (also referred to as Relying Party) cannot prove to a third party that a particular credential presentation took place. In the authenticated-channel architecture, the symmetric session key shared between chip and verifier means the verifier could itself have produced any authentication tag, so the transcript carries no transferable conviction. A persistent issuer signature has no such property: anyone in possession of the signed presentation can independently verify the issuer's endorsement, and the resulting transcripts therefore become attractive targets for data theft and unauthorised trading [29].

This framing is misleading though: The absence of deniability is not an inherent property of signature-based credentials, but a property of how they are presented. In particular, well understood cryptographic techniques allow transforming publicly verifiable signatures into designated-verifier forms at presentation time, ensuring that transcripts remain convincing to the intended verifier but are non-transferable to third parties.

Designated Verifier Signatures. These techniques are known as *Universal Designated Verifier Signatures* (UDVS), introduced by Steinfeld et al. [26] as a generalisation of the designated verifier signatures of Jakobsson, Sako, and Impagliazzo [21]. Here the verifier holds its own key pair. Using only the verifier's public key, the holder of a valid standard signature can transform it at presentation time into a *designated-verifier* signature. Such a signature convinces the intended verifier but is non-transferable to anyone else, since the verifier could have produced an indistinguishable one itself using its secret key.

One generic recipe for turning a standard signature into a UDVS uses a Σ-protocol OR-proof in the style of Cramer, Damgård, and Schoenmakers (CDS) [11]: the holder proves, in zero knowledge, knowledge of either a valid signature or the verifier's secret key. Since the verifier knows their own key, they could have produced the proof themselves. Tso et al. [27] showed nearly two decades ago that this recipe is instantiable for ECDSA using the sequential OR-proof technique of Abe, Ohkubo, and Suzuki (AOS) [1]. Interestingly, the construction of Tso et al. has received little follow-up (4 citations according to Google Scholar), possibly for lack of a concrete deployment demanding it. The EUDI-Wallet debate provides such an application, and our contribution is to restate the construction in a slightly simplified form and place it in this context.

Our Contributions. This paper revisits the authenticated channel vs. signed data framing as a question about the deniability of presentations. We show that

deniability is achievable for ECDSA-signed credentials as they are currently used in the EUDI-Wallet, without changing the signature scheme, the credential format, or the issuer-side PKI. Our contribution is not a new cryptographic primitive. Rather, we demonstrate that well-established techniques – largely absent from the current policy discussion – provide a practical path to reconcile signed credentials with potential deniability demands in the EUDI context. This yields a stronger privacy property than naïve signed presentations, while preserving the deployment advantages of the signed-data approach. Concretely, the paper makes the following contributions.

Revisiting the authenticated channel vs. signed data debate. We argue that the perceived trade-off between authenticated channels and signed credentials is not intrinsic to the choice of cryptographic primitive, but a property of how that primitive is exposed in the presentation protocol. Whether a deployment exposes the raw issuer signature or wraps it in a designated form is a design choice. We use this lens to discuss the suitability of designated verifier techniques, the relation to full-fledged zero-knowledge presentations (anonymous credentials), where deniability is an orthogonal rather than a free property, and discuss the limitations of cryptographic deniability against operational data-retention practices and broader real-world threat models.

ECDSA-compatible deniable presentations. We present a simple UDVS construction for ECDSA signatures based on a classic OR-proof over the native verification equation. Our construction is based on the observation that the ECDSA verification equation itself directly induces a discrete-logarithm relation suitable for use in an OR-proof. For a signature $\sigma = (r, s)$ on a message m under public key Q_s, verification reduces to computing $R = s^{-1}(\mathcal{H}(m)\,G + r\,Q_s)$ and checking $r \equiv x_R \bmod q$. Writing $\tilde{G} := \mathcal{H}(m)\,G + r\,Q_s$, a valid signature satisfies $R = s^{-1}\,\tilde{G}$ – a textbook Schnorr relation over base $\tilde{G}$. We compose a Schnorr proof of knowledge of s^{-1} with a Schnorr proof of knowledge of the verifier's secret key via the parallel CDS OR-technique [11], yielding a secure and lightweight UDVS scheme for ECDSA signatures. The construction is conceptually simpler than the AOS-based variant of [27], albeit generates slightly larger presentations. As the goal of our work is to convey the concept of deniability in the context of the EUDI wallet, we opted for the simpler construction, but also re-state the scheme by Tso et al. for completeness.

We show that our simple UDVS scheme produces a designated-verifier signature whose unconditional non-transferability follows by a straightforward simulator argument, and unforgeability follows from security of ECDSA and the DL assumption. Interestingly, the single-show setting already assumed in current EUDI deployments, where every credential is used only a single time for the purpose of Relying Party unlinkability, yields the implicit assumptions made in the UDVS security model. We also identify multi-show deniable presentation of a single ECDSA credential as an interesting open problem.

Outline. Section 2 discusses the authenticated-channel vs. signed-data debate, formalises what plausible deniability requires, and clarifies its limitations. Section 3 introduces the needed building blocks, in particular the UDVS syntax and security properties. Section 4 presents the ECDSA-based construction, its security analysis and discussion how it fits within the EUDI Wallet context.

2 Authenticated Channels, Signed Credentials, and Deniability

We start by revisiting the authenticated channel vs. signed data debate. We then define the deniability property at stake and describe the known cryptographic techniques for digital signatures. Finally, we clarify the relation to anonymous credentials, and discuss the limitations of cryptographic deniability.

2.1 The Authenticated Channel Approach (German eID)

The authenticated channel approach is the established mechanism of the German electronic identity card (*Personalausweis*) with eID functionality, specified in the BSI Technical Guideline TR-03110 [7,6]. The eID chip is provisioned at issuance with the citizen's attributes and a static, certified chip authentication key pair. At presentation, the chip and the Relying Party authenticate each other in an interactive session and derive a shared ephemeral symmetric session key; the chip's authenticity is thereby established implicitly, as only a chip holding the genuine private key can derive the correct key. The attributes are then read out over the resulting channel, each message authenticated by an HMAC under the shared key [7], giving the Relying Party both the attributes and assurance that they originate from a genuine, issuer-certified chip.

The combination – symmetric authentication established via ephemeral key agreement – is what gives the German eID its plausible deniability: Because the session key is symmetric, the Relying Party could itself compute any of the MAC tags it receives, so a third party given the transcript cannot tell a tag generated inside the chip from one fabricated by the Relying Party after the fact.

In the classic eID, this is a two-party interaction between chip and Relying Party, with no issuer involvement once the chip is provisioned. Adapting this approach to the more open EUDI-Wallet ecosystem, however, appears more challenging. Three variants have been considered [9], two of which have the PID Provider create the credential *on demand* at presentation, requiring an online issuer that participates in every presentation. The third provisions the credential into a secure element on the user's smartphone, recovering offline, issuer-free presentation, though this comes with its own challenges around securing such an approach at scale.

2.2 The Signed Data Approach

The signed-data approach pursued by the EUDI-Wallet [15] takes a different route: at issuance, the issuer produces a publicly verifiable ECDSA signature

over (a salted-hash commitment of) the user's attributes, which the wallet stores and forwards to a Relying Party at presentation time. Verification reduces to checking the issuer's signature against its certified public key.

A core advantage of the signed approach is that it decouples issuance from presentation: the issuer signs the credential once, and the wallet can subsequently present it offline, as a two-party interaction with the Relying Party, without the per-presentation issuer involvement that two of the three EUDI authenticated-channel variants require. The signed approach is also, however, the source of the deniability concern. The ECDSA signature is a persistent, publicly verifiable object: anyone in possession of it can independently verify the issuer's endorsement, without further interaction with either the issuer or the holder. A Relying Party, or an attacker who compromises its systems, could therefore prove to arbitrary third parties that authentic identity data was obtained, raising the concern that the resulting transcripts would become attractive targets for data theft and unauthorised trading [29]. This lack of plausible deniability is a central concern raised against the signed data approach.

2.3 Cryptographic Approaches to Deniability

Contrary to what the policy debate may suggest, the tension between signature-based authenticity and plausible deniability is neither new nor unsolved. To see why, recall what deniability formally requires. A credential presentation is *deniable* if there exists an efficient algorithm – a *simulator* – that the verifier itself can run, using its own secret key, to produce transcripts indistinguishable from genuine ones. If the verifier can produce the entire transcript on its own, no third party can tell whether a given transcript arose from a real interaction with the holder or was simply fabricated by the verifier; the transcript thus carries no transferable conviction. We refer to this property interchangeably as deniability or *non-transferability*.

This is exactly the guarantee the authenticated channel of the German eID provides: the symmetric session key known to the verifier is the secret that enables simulation and any transcript it could authenticate, the verifier could equally have produced alone. The same guarantee, however, is also achievable when the underlying authentication is a publicly verifiable signature, through *designated verifier signatures* (DVS), introduced by Jakobsson, Sako, and Impagliazzo [21], and their non-interactive generalisation, *Universal Designated Verifier Signatures* (UDVS) [26]. A UDVS lets the holder of a valid standard signature transform it, using only the verifier's public key, into a DV-signature that convinces the intended verifier but is non-transferable: the verifier's own secret key lets it simulate indistinguishable DV-signatures. This decouples issuance from presentation – the issuer signs once, and the holder presents deniably to any verifier without further issuer involvement [4] – so the choice of signed credentials over authenticated channels does not, in itself, preclude deniability.

One generic recipe [11] turns a standard signature into a UDVS via a Σ-protocol OR-proof: the holder proves, in zero knowledge, knowledge of *either* a valid signature *or* the verifier's secret key. Since the verifier knows their own

key, they could have produced the proof themselves – exactly the simulability that yields non-transferability. Tso et al. [27] instantiated this recipe for ECDSA already in 2007, but the construction has attracted hardly any follow-up. We revisit and simplify it in Section 4.

A complementary approach achieves the same non-transferability goal without any verifier key pair, by making the presentation an interactive zero-knowledge proof of possession of the issuer signature [28]. In this framework, formalised as an offline non-transferable authentication protocol (ONTAP)[23], non-transferability follows directly from the zero-knowledge simulator. While this can be instantiated for RSA and other standard signatures (including (EC)DSA) using Σ-protocols [28], it strictly requires interaction,as a non-interactive Fiat–Shamir transform would yield a transferable transcript. In our setting, where Relying Parties already hold certified keys, the designated-verifier route is preferable as it avoids additional rounds of interaction.

2.4 Relation to Anonymous Credentials

A parallel line of work seeks to add privacy to ECDSA-signed credentials via general-purpose zero-knowledge proofs of signature knowledge, with the Longfellow construction [19] as the most prominent example, alongside the BBS-based agenda for the EUDI-Wallet [22,2]. Such anonymous credential constructions provide selective disclosure and multi-show unlinkability across presentations – even when the issuer is corrupt. These solutions are particularly valuable in the *minimal-disclosure* regime, e.g., proving age-over-18 from a full identity credential. Current endeavors to integrate anonymous credentials are orthogonal to the deniability property pursued in this work: a zero-knowledge proof of knowledge of an issuer signature is convincing to its verifier and – in its standard non-interactive instantiation – transferable to third parties. Deniability is therefore an additional property that must be built on top of an anonymous credential typically through the same designated-verifier techniques we apply here. Further, there are two main aspects in which anonymous credentials and deniable signature techniques (on top of standard signatures) differ:

Deployment complexity. Real-world deployment of anonymous credentials within the EUDI framework remains an active engineering effort. Anonymous credentials either require a non-standard signature scheme such as BBS (in which case the zero-knowledge proof is a simple Schnorr proof), or a highly complex circuit-based zero-knowledge proof (in which case issuers can continue to use ECDSA). In contrast, the technique needed to add deniability is just a simple Schnorr proof [20] on top of a standard ECDSA signature.

Minimal vs. maximal disclosure. More fundamentally, anonymous credentials and deniability target different use cases. Anonymous credentials address the *minimal-disclosure* regime, in which the wallet reveals as little as possible. The deniability concern raised in [29], by contrast, is most acute in the *maximal-disclosure* regime – opening a bank account, hotel check-in, or KYC procedures

– where the wallet necessarily reveals most or all of a credential's attributes. In such transactions, anonymous credential techniques offer little protection: there are no attributes left to hide. The privacy concern is no longer *what* is revealed, but *to whom*, and whether the disclosed information can subsequently be transferred, sold, or compelled out of the verifier's records. This is precisely the regime in which plausible deniability matters most, and in which a deniable ECDSA presentation – preserving the wallet's existing credential format and signature primitive – is the right tool. The constructions presented here are therefore neither a substitute for nor a competitor of ZK-based anonymous credentials; they address an orthogonal threat model with strictly lighter cryptographic machinery.

2.5 Limitations of Deniability

Cryptographic deniability is a structural property of the presentation transcript, not a guarantee about the wider system in which that transcript is generated, stored, and interpreted. Two limitations, in particular, sit outside what any cryptographic construction can deliver, and frame what our result – and the deniability debate more broadly – actually achieve.

Operational data lifecycles in regulated sectors. The deniability property reasons about what a Relying Party can prove from the cryptographic transcript. In several motivating use cases – notably banking and other KYC-regulated sectors – the transcript is not the long-term form in which disclosed attributes are retained. Anti-money-laundering regulation (in the EU, the AMLD [16]) requires retention of verification evidence for several years but is silent on its cryptographic form. In practice, identity attributes are extracted from the cryptographic envelope at verification time and committed to the institution's internal database under its own data-integrity controls. A Relying Party wishing to convince a third party of a past presentation can therefore rely on internally authenticated records whose evidentiary value derives from the institution's auditing regime, not from the cryptographic transcript.

This narrows the threat model rather than undermining the case for deniability. Cryptographic deniability protects against leakage or sale of the original presentation transcript beyond the verifier's institutional perimeter. It does not protect against an institution that explicitly authenticates the attributes internally in line with its retention obligations, nor against settings in which contextual evidence alone suffices to convince a data broker of the attributes' authenticity.

Cryptographic deniability vs. real-world deniability. A second, deeper limitation concerns the gap between deniability as a cryptographic notion and deniability as a property that holds up in legal, social, and adversarial real-world settings. Reitinger et al. [25] and Yadav et al. [31] find, through survey and case-study evidence, that simple denials of message authorship are not, on their own, persuasive

to lay observers or courts. Collins et al. [10] formalise the gap in a model of *real-world deniability* that incorporates the wider system context (server logs, device state, account metadata) and show that systems with strong cryptographic deniability claims – Signal and DKIM-protected email – fail to provide deniability under their model; in their analysis of 140 Swiss court cases involving WhatsApp transcripts, judges consistently accept such evidence even when disputed, and deniability is not mentioned or considered in any case. The lesson is that the property our construction delivers is necessary but not sufficient for the practical privacy outcome motivating the debate. A Relying Party with logs of which wallet authenticated when, or trusted to store only verified data or verifying presentations inside tamper-resistant, attested hardware [28], can corroborate a disclosed-attribute claim against a third party even when the underlying signature is deniable. The cryptographic property is a structural prerequisite: without it, the transcript itself constitutes transferable evidence that no operational mitigation can retroactively remove. But it must be complemented by operational and architectural choices to translate into deniability in the sense the policy debate cares about, and whether such measures can fully close the gap between cryptographic and real-world deniability remains an open question.

3 Preliminaries

This section recalls the two cryptographic primitives that the construction in Section 4 composes: the ECDSA signature scheme, which is the used signature primitive for the EUDI-Wallet, and the Universal Designated Verifier Signature (UDVS), which is the abstract object our construction realises.

3.1 ECDSA Signatures

Definition 1 (ECDSA). *Let E be an elliptic curve, G be a generator of a cyclic subgroup of prime order q, and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ a hash function. ECDSA is the signature scheme defined by the following algorithms.*

$\mathsf{Setup}(1^\lambda)$**:** *Generates curve parameters (E, G, q) for security parameter λ and fixes H. Outputs $\mathsf{pp} := (E, G, q, \mathsf{H})$.*

$\mathsf{KeyGen}(\mathsf{pp})$**:** *Samples $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q^*$, sets $\mathsf{pk} := \mathsf{sk} \cdot G$, and outputs $(\mathsf{sk}, \mathsf{pk})$.*

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *Samples a per-signature randomness $k \xleftarrow{\$} \mathbb{Z}_q^*$, computes $R := kG$ and $r := x_R \bmod q$, then $s := k^{-1}(\mathsf{H}(m) + r \cdot \mathsf{sk}) \bmod q$. (If $r = 0$ or $s = 0$, the signer restarts with fresh k.) Outputs $\sigma := (r, s)$.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** *On input $\sigma = (r, s)$, rejects unless $1 \le r, s \le q - 1$. Otherwise computes $R := s^{-1}\big(\mathsf{H}(m)\, G + r \cdot \mathsf{pk}\big)$ and accepts iff $r \equiv x_R \bmod q$.*

Remark. For the construction in Section 4 it is useful to rewrite verification in the form $R = s^{-1}\widetilde{G}$, where $\widetilde{G} := \mathsf{H}(m)\, G + r \cdot \mathsf{pk}$. A valid ECDSA signature thus exhibits a discrete-logarithm relation between two publicly computable points R and $\widetilde{G}$, with the multiplicative inverse s^{-1} as witness. This is the algebraic structure we exploit.

3.2 Universal Designated Verifier Signatures (UDVS)

A UDVS scheme involves three parties: a *signer* S, who issues an ordinary publicly verifiable signature; a *holder* H, who receives this signature from S and later designates it; and a *designated verifier* V, to whom the holder presents the signature in a non-transferable form. The signer is uninvolved at presentation time; the holder uses only the verifier's public key to designate. The role assignment maps onto the EUDI-Wallet as follows: S is the credential issuer, H is the wallet, and V is the Relying Party.

Definition 2 (Universal Designated Verifier Signatures (UDVS)). *A UDVS scheme consists of the following PPT algorithms.*

$\mathsf{Setup}(1^\lambda)$**:** *outputs public parameters* pp *shared by all parties.*

$\mathsf{SKeyGen}(\mathsf{pp})$**:** *generates the signer's key pair* $(\mathsf{pk}_S, \mathsf{sk}_S)$*. In the EUDI setting this is the issuer's key pair.*

$\mathsf{VKeyGen}(\mathsf{pp})$**:** *generates the designated verifier's key pair* $(\mathsf{pk}_V, \mathsf{sk}_V)$*. The holder will need only* pk_V *at presentation time;* sk_V *enables the verifier's simulation capability that underlies non-transferability.*

$\mathsf{PSign}(\mathsf{sk}_S, m)$**:** *produces an* ordinary *publicly verifiable signature* σ *on* m*.*

$\mathsf{PVerify}(\mathsf{pk}_S, m, \sigma)$**:** *verifies an ordinary signature against the signer's public key; outputs* $b \in \{0, 1\}$*.*

$\mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$**:** *is the holder's presentation algorithm. It transforms the ordinary signature* σ *into a* designated verifier signature *(DV)* $\widehat{\sigma}$ *for verifier* pk_V*. The output convinces* V *but is non-transferable to any third party.*

$\mathsf{Sim}(\mathsf{pk}_S, \mathsf{sk}_V, m)$**:** *is the simulator that uses the verifier's secret key to produce a designated verifier signature on* m without *an underlying ordinary signature. Its existence is what makes* $\widehat{\sigma}$ *non-transferable: any transcript could have been produced by* V *alone.*

$\mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \widehat{\sigma})$**:** *verifies a designated verifier signature; outputs* $b \in \{0, 1\}$*.*

Correctness *requires that for all honestly generated keys and all messages* m*:*

$$\begin{aligned} \mathsf{PVerify}(\mathsf{pk}_S, m, \mathsf{PSign}(\mathsf{sk}_S, m)) &= 1, \\ \mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \mathsf{PSign}(\mathsf{sk}_S, m))) &= 1, \\ \mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \mathsf{Sim}(\mathsf{pk}_S, \mathsf{sk}_V, m)) &= 1. \end{aligned}$$

The third correctness condition is the structural reason for the deniability property: the verifier's secret key alone suffices to produce designated signatures that pass $\mathsf{DVerify}$, which is what enables the simulation argument underlying non-transferability.

Security Properties A UDVS scheme is required to provide two security properties: unforgeability of the designated verifier signature, and non-transferability of the conviction it provides to the verifier.

DVS-unforgeability. Two notions of unforgeability arise in UDVS schemes. *OS-unforgeability* is the standard EUF-CMA notion for the underlying signature PSign. *DVS-unforgeability* is the stronger requirement that an adversary, even one that learns ordinary signatures on arbitrary messages of its choice, cannot produce a designated verifier signature on a fresh message that passes $\mathsf{DVerify}$. As noted in [26], DVS-unforgeability implies OS-unforgeability, since any OS-forgery can be designated to yield a DVS-forgery via the public algorithm DSign. Thus, it is sufficient to focus on DVS-unforgeability for the remainder of this work.

Definition 3 (DVS-unforgeability [24]). *Let* $\mathsf{S} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{PSign}, \mathsf{PVerify}, \mathsf{DSign}, \mathsf{Sim}, \mathsf{DVerify})$ *be a UDVS scheme. Consider the following experiment between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*.*

Setup. $\mathcal{C}$ *runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{SKeyGen}(\mathsf{pp})$*, and* $(\mathsf{pk}_V, \mathsf{sk}_V) \leftarrow \mathsf{VKeyGen}(\mathsf{pp})$*. It gives* $(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_V)$ *to* $\mathcal{A}$*.*

Queries. $\mathcal{A}$ *adaptively accesses the following oracles:*

- $\mathcal{O}_{\mathsf{PSign}}(m)$*: returns* $\sigma \leftarrow \mathsf{PSign}(\mathsf{sk}_S, m)$*.*
- $\mathcal{O}_{\mathsf{DSign}}(m)$*: returns* $\widehat{\sigma} \leftarrow \mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$*, where* σ *is generated honestly if not already defined.*
- $\mathcal{O}_{\mathsf{DVerify}}(m, \widehat{\sigma})$*: returns* $\mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \widehat{\sigma})$*.*

Forgery. $\mathcal{A}$ *outputs* $(m^*, \widehat{\sigma}^*)$*.*

$\mathcal{A}$ *wins if* $\mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m^*, \widehat{\sigma}^*) = 1$ *and* m^* *was queried to neither* $\mathcal{O}_{\mathsf{PSign}}$ *nor* $\mathcal{O}_{\mathsf{DSign}}$*. The scheme provides DVS-unforgeability if for every PPT* $\mathcal{A}$ *the winning probability is negligible in* λ*.*

Non-transferability. Non-transferability captures the property that a designated verifier signature, while convincing to the verifier, conveys no transferable evidence to any third party. Formally, this is the requirement that designated signatures produced by an honest holder are indistinguishable from those produced by the simulator Sim using only the verifier's secret key. If the two distributions coincide, then any transcript the holder could produce can also be produced by the verifier acting alone, so a third party observing a transcript cannot distinguish the two cases.

Definition 4 (Non-transferability). *Let* S *be a UDVS scheme. Consider the following experiment between a challenger* $\mathcal{C}$ *and a PPT distinguisher* $\mathcal{D}$*.*

Setup. $\mathcal{C}$ *runs* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{SKeyGen}(\mathsf{pp})$*, and* $(\mathsf{pk}_V, \mathsf{sk}_V) \leftarrow \mathsf{VKeyGen}(\mathsf{pp})$*. It gives* $(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_V, \mathsf{sk}_V)$ *to* $\mathcal{D}$*.*

Challenge. $\mathcal{D}$ *outputs a message* m^**.* $\mathcal{C}$ *samples* $b \xleftarrow{\$} \{0,1\}$ *and computes*

$$\widehat{\sigma}^* \leftarrow \begin{cases} \mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m^*, \mathsf{PSign}(\mathsf{sk}_S, m^*)) & \text{if } b = 0, \\ \mathsf{Sim}(\mathsf{pk}_S, \mathsf{sk}_V, m^*) & \text{if } b = 1. \end{cases}$$

$\mathcal{C}$ *returns* $\widehat{\sigma}^*$ *to* $\mathcal{D}$*.*

Guess. $\mathcal{D}$ *outputs a bit* b' *and wins if* $b' = b$*.*

The scheme provides non-transferability *if for every PPT $\mathcal{D}$ the advantage* $\left|\Pr[b' = b] - \frac{1}{2}\right|$ *is negligible in λ. We say the scheme provides* unconditional non-transferability *if the two distributions are identical, in which case the advantage is zero for every (computationally unbounded) $\mathcal{D}$.*

4 Deniable ECDSA Credentials

We now show how deniability can be added on top of standard ECDSA signatures. Recall from the remark following Definition 1 that writing $\widetilde{G} := \mathsf{H}(m)\, G + r \cdot \mathsf{pk}_S$, a valid ECDSA signature satisfies

$$R \;=\; s^{-1} \cdot \widetilde{G},$$

which is a textbook Schnorr relation over base $\widetilde{G}$. A holder in possession of (r, s) can therefore prove knowledge of the scalar s^{-1} by a standard Σ-protocol. To obtain a designated-verifier signature, we compose this proof, via the CDS OR-technique [11], with a Schnorr proof of knowledge of the verifier's secret key sk_V satisfying $\mathsf{pk}_V = \mathsf{sk}_V \cdot G$. The Fiat–Shamir-compiled result convinces the verifier that the holder possesses a valid ECDSA signature *or* that the verifier itself produced the proof using sk_V; since the verifier knows sk_V, the proof is non-transferable. The construction is given in detail in Section 4.1.

This yields the same security property as the AOS-based scheme of Tso et al. [27] – unconditional non-transferability – but is built from a single ingredient (CDS OR-composition of two Schnorr proofs) and works directly with the ECDSA verification equation, without the auxiliary witness re-randomisation $h = s/t$ introduced in [27]. In Section 4.2, we present an alternative construction that likewise eliminates the re-randomisation step but replaces the parallel CDS OR-proof with the more compact sequential AOS OR-proof [1]. The resulting scheme improves upon [27] in both simplicity and signature size, while being more compact than the CDS variant at the cost of slightly more conceptual complexity.

Security model and EUDI fit. The construction is analysed in the UDVS security model of Section 3: DVS-unforgeability (Definition 3) and non-transferability against a malicious verifier possessing sk_V (Definition 4). Two aspects of this model deserve emphasis in the EUDI context. First, the non-transferability distinguisher is *not* given the underlying signature $\sigma = (r, s)$, matching the EUDI deployment reality: the issuer's signature lives inside the wallet and is never transmitted to a Relying Party, which only ever sees designated forms. Second, the model assumes a single presentation per credential, consistent with how the EUDI-Wallet currently obtains Relying-Party unlinkability for ECDSA-signed credentials in the absence of anonymous credentials [15,8]: each ECDSA-signed credential is used at most once. Section 4.3 returns to multi-show deniability and to the role of the wallet's holder-binding signature.

4.1 Parallel OR-Proof Construction

We instantiate the UDVS algorithms of Definition 2 in the standard ECDSA setting of Section 3, additionally using a Fiat–Shamir hash $\mathsf{H}_{\mathrm{FS}} : \{0,1\}^* \to \mathbb{Z}_q$.

Setup, Key Generation, and Issuance. The instantiations of Setup, $\mathsf{SKeyGen}$, $\mathsf{VKeyGen}$, PSign, and $\mathsf{PVerify}$ are essentially standard ECDSA; we summarise them here for completeness and briefly discuss their EUDI-specific aspects and verifier key management.

$\mathsf{Setup}(1^\lambda)$**:** outputs $\mathsf{pp} = (E, G, q, \mathsf{H}, \mathsf{H}_{\mathrm{FS}})$, i.e. the ECDSA setup of Definition 1 extended with the Fiat–Shamir hash H_{FS}.

$\mathsf{SKeyGen}(\mathsf{pp})$**:** standard ECDSA key generation: sample $\mathsf{sk}_S \xleftarrow{\$} \mathbb{Z}_q^*$ and output $\mathsf{pk}_S := \mathsf{sk}_S \cdot G$.

$\mathsf{VKeyGen}(\mathsf{pp})$**:** DL-based key generation on the same curve: sample $\mathsf{sk}_V \xleftarrow{\$} \mathbb{Z}_q^*$ and output $\mathsf{pk}_V := \mathsf{sk}_V \cdot G$.

$\mathsf{PSign}(\mathsf{sk}_S, m), \mathsf{PVerify}(\mathsf{pk}_S, m, \sigma)$**:** standard ECDSA signing and verification on m exactly as in Definition 1.

Issuance. The issuer produces $\sigma = (r, s)$ on a message m, which in the SD-JWT setting is the salted-hash commitment to the user's attributes. The wallet stores (m, σ) together with a holder-controlled binding key pair $(\mathsf{sk}_H, \mathsf{pk}_H)$ that is part of the issued credential and supports the Key Binding JWT (KB-JWT) [18] produced at presentation time. We treat m as an opaque bit string; the holder binding mechanism and its interaction with deniability are discussed in Section 4.3. No interaction with the issuer is required at presentation time.

Verifier key management. At the start of a presentation session, the verifier sends pk_V to the holder. Crucially, the holder must be assured that the verifier indeed knows the corresponding secret key sk_V, as this is what allows the simulator-based deniability argument to go through. This can be handled in two ways: either the verifier proves possession of sk_V at the time of key advertisement, or the certificate authority issuing pk_V already attests to such possession at registration time. Notably, this requirement integrates cleanly into the EUDI architecture: Relying Parties must in any case authenticate themselves to the wallet using a certified key pair – e.g., the certified key in the TLS server certificate or the EUDI Relying-Party certificate – so $(\mathsf{sk}_V, \mathsf{pk}_V)$ can simply be that same key pair, and no additional verifier-side key management is required.

Designation: $\mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$. The holder, given (m, r, s) and pk_V, proceeds as follows.

Step 1: Recompute the verification point. Compute

$$\widetilde{G} \;:=\; \mathsf{H}(m) \cdot G + r \cdot \mathsf{pk}_S, \qquad R \;:=\; s^{-1} \cdot \widetilde{G}.$$

By construction, $R = s^{-1}\widetilde{G}$, and the holder knows the witness $w := s^{-1} \bmod q$.

Step 2: CDS OR-proof. The holder produces a non-interactive proof of the statement

$$\{\, \exists w : R = w \cdot \widetilde{G} \,\} \;\vee\; \{\, \exists \mathsf{sk}_V : \mathsf{pk}_V = \mathsf{sk}_V \cdot G \,\},$$

using the CDS parallel OR-composition of two Schnorr Σ-protocols [11]. The holder knows the witness for the left clause and simulates the right:

1. Pick $k_L \xleftarrow{\$} \mathbb{Z}_q$, set $A_L := k_L \cdot \widetilde{G}$. (real, left)
2. Pick $z_R, c_R \xleftarrow{\$} \mathbb{Z}_q$, set $A_R := z_R \cdot G - c_R \cdot \mathsf{pk}_V$. (sim., right)
3. Compute $c := \mathsf{H}_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_L, A_R, m)$.
4. Set $c_L := c - c_R \bmod q$ and $z_L := k_L + c_L \cdot w \bmod q$. (real, left)

Step 3: Output. The DV-signature is $\widehat{\sigma} := (R, A_L, A_R, c_L, c_R, z_L, z_R)$.

Transcript size. A size-optimised encoding transmits only one of $\{c_L, c_R\}$, since the other follows from $c_L + c_R = c$. The DV-signature thus comprises three group elements and three scalars; for curves at the 128-bit security level (e.g., P-256), about $3 \times 33 + 3 \times 32 = 195$ bytes.

Session freshness. Credential presentations are typically bound to a verifier-chosen session nonce to ensure freshness and prevent replay of a presentation across sessions. In the EUDI-Wallet, this responsibility is delegated entirely to the Key Binding JWT [18], which contains the verifier's nonce and is signed by the holder's binding key. The DV-signature constructed above therefore does not need to incorporate a session nonce of its own. The deniability implications of this holder-binding signature are discussed in Section 4.3.

Designated Verification: $\mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \widehat{\sigma})$. The verifier parses $\widehat{\sigma}$ as $(R, A_L, A_R, c_L, c_R, z_L, z_R)$. First, it recomputes the verification base point: $r' := x_R \bmod q$ and $\widetilde{G} := \mathsf{H}(m) \cdot G + r' \cdot \mathsf{pk}_S$ (note that r' is recovered from R, mirroring standard ECDSA verification). Second, it recomputes $c := \mathsf{H}_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_L, A_R, m)$. Finally, it accepts iff

$$c = c_L + c_R \bmod q, \quad z_L \cdot \widetilde{G} = A_L + c_L \cdot R, \quad \text{and} \quad z_R \cdot G = A_R + c_R \cdot \mathsf{pk}_V.$$

Security and Deniability. We now show that the above construction achieves non-transferability (Definition 4) and DVS-unforgeability (Definition 3).

High-level intuition. Non-transferability follows from the honest-verifier zero-knowledge property of CDS OR-proofs: knowing sk_V, the verifier can produce transcripts whose distribution is identical to honest holder-produced transcripts. DVS-unforgeability follows from the special-soundness of CDS OR-proofs combined with the EUF-CMA security of ECDSA: an adversary producing a valid DV-signature on a fresh message must, via standard rewinding, yield either an ECDSA forgery on that message or the verifier's secret key.

On the role of R in the transcript. Note that the transcript reveals R in the clear. Since $r = x_R \bmod q$, this implicitly reveals the first component of the underlying ECDSA signature. Thus, $\widehat{\sigma}$ is not a zero-knowledge presentation of σ, but this is not needed here, and does not harm the deniability goal. First, by Definition 4, the distinguisher is not given $\sigma = (r, s)$, and R alone does not allow reconstruction of the signature (which requires both r and s). Second, R is uniformly distributed in $\langle G \rangle$ over the randomness of ECDSA signing, so it leaks no information about the issuer's secret key or the message. Third, even an adversary who learns r through another channel cannot link two presentations of *distinct* credentials, since each credential is signed under fresh per-signature randomness. The leakage is structural: it is the price paid for staying within the standard ECDSA verification equation rather than committing to R inside a more expensive proof. Two presentations of the *same* credential are more subtle; see Section 4.3.

The simulator. The simulator $\mathsf{Sim}(\mathsf{pk}_S, \mathsf{sk}_V, m)$ proceeds as follows:

1. Pick $R \xleftarrow{\$} \langle G \rangle$ uniformly at random.
2. Compute $\widetilde{G} := \mathsf{H}(m) \cdot G + (x_R \bmod q) \cdot \mathsf{pk}_S$.
3. Pick $k_R \xleftarrow{\$} \mathbb{Z}_q$, set $A_R := k_R \cdot G$. (real, right)
4. Pick $z_L, c_L \xleftarrow{\$} \mathbb{Z}_q$, set $A_L := z_L \cdot \widetilde{G} - c_L \cdot R$. (sim., left)
5. Compute $c := \mathsf{H}_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_L, A_R, m)$, set $c_R := c - c_L \bmod q$ and $z_R := k_R + c_R \cdot \mathsf{sk}_V \bmod q$. (real, right)

Output $\widehat{\sigma} := (R, A_L, A_R, c_L, c_R, z_L, z_R)$. The simulator runs the OR-proof with the clause roles swapped: it executes the right clause honestly with the witness sk_V and simulates the left, mirroring the honest holder's procedure of Section 4.1.

Lemma 1 (Simulator indistinguishability). *For any (m, pk_V), the distribution of DV-signatures produced by DSign is identical to that produced by Sim, provided the distinguisher does not observe the underlying signature (r, s).*

Proof. Let $\widehat{\sigma}_{\mathsf{DS}} \leftarrow \mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$ and $\widehat{\sigma}_{\mathsf{Sim}} \leftarrow \mathsf{Sim}(\mathsf{pk}_S, \mathsf{sk}_V, m)$. We show that the two output distributions are identical up to negligible statistical distance.

In DSign, the point $R = kG$ for the ECDSA nonce $k \xleftarrow{\$} \mathbb{Z}_q^*$ is uniform over $\langle G \rangle \setminus \{O\}$. In Sim, R is sampled uniformly from $\langle G \rangle$. The statistical distance between these marginals is $1/q$, which is negligible.

It remains to show that, conditioned on any fixed $R^\star \in \langle G \rangle \setminus \{O\}$, both algorithms induce the same distribution on the proof components. Let $\widetilde{G}^\star := \mathsf{H}(m) \cdot G + (x_{R^\star} \bmod q) \cdot \mathsf{pk}_S$. Conditioned on $R = R^\star$, both algorithms produce a CDS OR-proof for the pair of statements $(R^\star = w \cdot \widetilde{G}^\star) \vee (\mathsf{pk}_V = \mathsf{sk}_V \cdot G)$, where one branch is executed honestly with the known witness and the other is simulated. In both cases, the algorithm samples exactly three independent uniform values from $\mathbb{Z}_q$: the honest branch contributes a nonce (k_L resp. k_R), and the simulated branch contributes a challenge and a response (c_R, z_R resp. c_L, z_L).

All remaining components of the tuple $(A_L, A_R, c_L, c_R, z_L, z_R)$ are uniquely determined by the verification equations

$$\begin{aligned} z_L \cdot \widetilde{G}^\star &= A_L + c_L \cdot R^\star, \\ z_R \cdot G &= A_R + c_R \cdot \mathsf{pk}_V, \\ c_L + c_R &= \mathsf{H}_{\mathrm{FS}}(R^\star, \widetilde{G}^\star, \mathsf{pk}_V, A_L, A_R, m). \end{aligned}$$

In particular, for any fixed accepting tuple, exactly one choice of the three random values produces it, so both algorithms assign probability $1/q^3$ to every such tuple. Since both the marginal on R and the conditional distribution of the proof given R coincide, the claim follows.

Theorem 1 (Non-transferability). *The construction provides unconditional non-transferability (Definition 4), provided the distinguisher does not observe the underlying ECDSA signature.*

Proof. Immediate from Lemma 1: any transcript produced by DSign could equally have been produced by Sim with identical distribution; a distinguisher therefore has advantage zero.

Theorem 2 (DVS-unforgeability). *If ECDSA is EUF-CMA-secure and the discrete logarithm problem in $\langle G \rangle$ is hard, then the construction satisfies DVS-unforgeability (Definition 3) in the random oracle model.*

Proof. The proof follows by a standard reduction argument: We reduce DVS-unforgeability to EUF-CMA security of ECDSA and hardness of the discrete logarithm problem in $\langle G \rangle$.

Setup. Let $\mathcal{B}$ be an adversary against DVS-unforgeability (Definition 3), making at most q_H queries to H_{FS}. We construct a reduction $\mathcal{A}$ that uses $\mathcal{B}$ as a black box to either forge ECDSA signatures or solve the discrete logarithm problem.

$\mathcal{A}$ receives a public key pk_S together with a signing oracle $\mathcal{O}_{\mathsf{ECDSA}}(\cdot)$ from an ECDSA EUF-CMA challenger, and receives a discrete-log instance $\mathsf{pk}_V \in \langle G \rangle$. Note that $\mathcal{A}$ knows neither the signing key sk_S nor the discrete logarithm $\mathsf{sk}_V = \log_G \mathsf{pk}_V$.

$\mathcal{A}$ sets $\mathsf{pp} := (E, G, q, \mathsf{H}, \mathsf{H}_{\mathrm{FS}})$, embeds pk_V as the designated verifier's public key, and simulates the DVS-unforgeability game for $\mathcal{B}$: it provides $(\mathsf{pp}, \mathsf{pk}_S, \mathsf{pk}_V)$ to $\mathcal{B}$ and answers oracle queries as follows.

Oracle simulation.

- H_{FS}*-oracle:* $\mathcal{A}$ maintains a list H_{FS}-List, initially empty. For each query on input $(R_i, \widetilde{G}_i, \mathsf{pk}_V, A_{L,i}, A_{R,i}, m_i)$: if the input already appears in H_{FS}-List, $\mathcal{A}$ returns the previously stored value; otherwise $\mathcal{A}$ samples $c_i \xleftarrow{\$} \mathbb{Z}_q$, stores the tuple $(R_i, \widetilde{G}_i, \mathsf{pk}_V, A_{L,i}, A_{R,i}, m_i, c_i)$ in H_{FS}-List, and returns c_i to $\mathcal{B}$.
- $\mathcal{O}_{\mathsf{PSign}}$*-queries:* Whenever $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{PSign}}(m)$, $\mathcal{A}$ forwards m to its own ECDSA signing oracle $\mathcal{O}_{\mathsf{ECDSA}}$ (for pk_S) and returns the resulting signature $\sigma = (r, s)$ to $\mathcal{B}$. The pair (m, σ) is stored in a sign-list $\mathcal{Q}_S$.

- $\mathcal{O}_{\mathsf{DSign}}$*-queries:* On input m, $\mathcal{A}$ first ensures that it knows an ordinary signature on m: if (m, σ) is not yet in $\mathcal{Q}_S$, it queries its ECDSA signing oracle $\mathcal{O}_{\mathsf{ECDSA}}$ to obtain $\sigma = (r, s)$ and stores (m, σ) in $\mathcal{Q}_S$. It then computes $\widehat{\sigma} \leftarrow \mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$ and returns $\widehat{\sigma}$ to $\mathcal{B}$.
- $\mathcal{O}_{\mathsf{DVerify}}$*-queries:* On input $(m, \widehat{\sigma})$, $\mathcal{A}$ runs $b \leftarrow \mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m, \widehat{\sigma})$ and returns b to $\mathcal{B}$.

Here, H_{FS} is modelled as a random oracle. The simulation is perfect: H_{FS}-responses are uniform and consistent, ECDSA signatures from $\mathcal{O}_{\mathsf{ECDSA}}$ are identically distributed to real signatures, DSign requires only σ and pk_V, not any secret key, and $\mathsf{DVerify}$ is publicly computable.

At the end of this interaction, $\mathcal{B}$ outputs a forgery $(m^*, \widehat{\sigma}^*)$ with

$$\widehat{\sigma}^* = (R^*, A_L^*, A_R^*, c_L^*, c_R^*, z_L^*, z_R^*).$$

By assumption, $\mathcal{B}$ wins the DVS-unforgeability game with non-negligible probability, i.e.,

$$\mathsf{DVerify}(\mathsf{pk}_S, \mathsf{pk}_V, m^*, \widehat{\sigma}^*) = 1$$

and m^* was never submitted to $\mathcal{O}_{\mathsf{PSign}}$ or $\mathcal{O}_{\mathsf{DSign}}$ and in particular $m^* \notin \mathcal{Q}_S$.

Applying the Forking Lemma. We apply the General Forking Lemma [3], where the idea is to run $\mathcal{B}$ twice with identical randomness but a fresh random oracle response at a critical point, yielding two accepting transcripts with distinct challenges from which a witness can be extracted.

In the *first run*, $\mathcal{A}$ simulates the environment as described above. At the end of this run, $\mathcal{B}$ outputs the successful forgery $\widehat{\sigma}^*$ on message m^*. By correctness of verification, the challenge c^* satisfies $c^* = \mathsf{H}_{\mathrm{FS}}(R^*, \widetilde{G}^*, \mathsf{pk}_V, A_L^*, A_R^*, m^*)$. With probability at least $1 - 1/q$, this value was obtained by $\mathcal{B}$ through an explicit query to the H_{FS}-oracle. Let i^* denote the index of this critical query.

In the *second run*, $\mathcal{A}$ replays $\mathcal{B}$ with the same random tapes and simulates all oracles identically, except: for any j-th H_{FS} query with $j < i^*$, $\mathcal{A}$ returns the same answer as in the first run; for any j-th H_{FS} query with $j \geq i^*$, $\mathcal{A}$ samples a fresh random value $c'_j \xleftarrow{\$} \mathbb{Z}_q$.

Since the views of $\mathcal{B}$ are identical up to the i^*-th H_{FS} query, the input to this critical query is the same in both runs. In particular,

$$R^* = R', \quad m^* = m', \quad \widetilde{G}^* = \widetilde{G}', \quad A_L^* = A'_L, \quad A_R^* = A'_R.$$

The responses, however, differ: the first run returns c^* and the second run returns $c' \neq c^*$ (with probability $1 - 1/q$).

By the General Forking Lemma [3], with non-negligible probability, both runs produce valid forgeries at the same critical query index i^*. The second run thus yields another valid forgery $\widehat{\sigma}' = (R^*, A_L^*, A_R^*, c'_L, c'_R, z'_L, z'_R)$ on m^*, with $c'_L + c'_R = c'$ and $c^* \neq c'$.

Witness extraction Since both forgeries pass verification, the verification equations hold:

$$z_R^* \cdot G = A_R^* + c_R^* \cdot \mathsf{pk}_V \tag{1}$$
$$z_R' \cdot G = A_R^* + c_R' \cdot \mathsf{pk}_V \tag{2}$$
$$z_L^* \cdot \widetilde{G}^* = A_L^* + c_L^* \cdot R^* \tag{3}$$
$$z_L' \cdot \widetilde{G}^* = A_L^* + c_L' \cdot R^* \tag{4}$$

From $c^* \neq c'$ and $c^* = c_L^* + c_R^*$, $c' = c_L' + c_R'$, at least one of $c_L^* \neq c_L'$ or $c_R^* \neq c_R'$ must hold.

Case 1: Suppose $c_R^* \neq c_R'$. Subtracting (2) from (1) yields

$$(z_R^* - z_R') \cdot G = (c_R^* - c_R') \cdot \mathsf{pk}_V.$$

Then $\mathsf{sk}_V = (z_R^* - z_R')/(c_R^* - c_R') \bmod q$ solves the discrete logarithm problem for (G, q, pk_V), contradicting the DL assumption, and $\mathcal{A}$ succeeds.

Case 2: Otherwise $c_R^* = c_R'$. Since $c^* \neq c'$, it follows that $c_L^* \neq c_L'$. Subtracting (4) from (3) yields

$$(z_L^* - z_L') \cdot \widetilde{G}^* = (c_L^* - c_L') \cdot R^*.$$

Then $\mathcal{A}$ extracts

$$w^* \;:=\; \frac{z_L^* - z_L'}{c_L^* - c_L'} \bmod q, \qquad \text{satisfying} \quad R^* = w^* \cdot \widetilde{G}^*.$$

Setting $s^* := (w^*)^{-1} \bmod q$ and $r^* := x_{R^*} \bmod q$, the pair (r^*, s^*) is a valid ECDSA signature on m^* under pk_S: since $\widetilde{G}^* = \mathsf{H}(m^*) \cdot G + r^* \cdot \mathsf{pk}_S$ and $R^* = (s^*)^{-1} \cdot \widetilde{G}^*$, the standard ECDSA verification equation is satisfied. $\mathcal{A}$ queries exclusively the ECDSA oracle $\mathcal{O}_{\mathsf{ECDSA}}$ when servicing $\mathcal{O}_{\mathsf{PSign}}(m)$ or $\mathcal{O}_{\mathsf{DSign}}(m)$ from $\mathcal{B}$; in both cases m is added to $\mathcal{Q}_S$. By the winning condition, $m^* \notin \mathcal{Q}_S$, so m^* was never submitted to $\mathcal{O}_{\mathsf{ECDSA}}$. Hence $(m^*, (r^*, s^*))$ constitutes a valid EUF-CMA forgery, contradicting the security of ECDSA.

Success probability. Let $\mathsf{Adv}_{\mathcal{B}}$ be the probability that $\mathcal{B}$ wins the DVS-unforgeability game. Since a valid forgery requires $c^* = \mathsf{H}_{\mathrm{FS}}(R^*, \widetilde{G}^*, \mathsf{pk}_V, A_L^*, A_R^*, m^*)$, and the probability that $\mathcal{B}$ produces this value without querying H_{FS} is at most $1/q$, the accepting probability (i.e., that $\mathcal{B}$ wins and the critical query index i^* is well-defined) satisfies $\mathsf{acc} \geq \mathsf{Adv}_{\mathcal{B}} - 1/q$. By the General Forking Lemma [3], the forking procedure succeeds with probability at least

$$\mathsf{frk} \;\geq\; \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{q_H} - \frac{1}{q} \right),$$

where q_H is the number of H_{FS}-queries. Since $\mathsf{Adv}_{\mathcal{B}}$ is non-negligible by assumption and q_H is polynomial in λ, frk is non-negligible. Whenever the forking

succeeds, $\mathcal{A}$ either solves the discrete logarithm problem (Case 1) or extracts an ECDSA forgery (Case 2). Therefore,

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{ECDSA}}(\mathcal{A}) \;+\; \mathsf{Adv}^{\mathsf{DL}}(\mathcal{A}) \;\geq\; \mathsf{frk},$$

contradicting the assumed hardness of at least one of the two problems. This completes the proof of Theorem 2.

4.2 Alternative: AOS Sequential OR-Proof Construction

The direct relation $R = s^{-1}\widetilde{G}$ also admits a more compact instantiation using the sequential OR-proof technique of Abe, Ohkubo, and Suzuki [1]. We present this variant below; it parallels Tso et al. [27] but operates directly on our simplified witness without their auxiliary re-randomisation. The instantiation of Setup, $\mathsf{SKeyGen}$, $\mathsf{VKeyGen}$, PSign, and $\mathsf{PVerify}$ is unchanged; only DSign, $\mathsf{DVerify}$, and Sim differ. We use two independent Fiat–Shamir hashes $\mathsf{H}_{\mathrm{FS}}, \mathsf{H}'_{\mathrm{FS}} : \{0,1\}^* \to \mathbb{Z}_q$ (in practice derived from a single hash function, modelled as random oracle by domain separation).

Designation. $\mathsf{DSign}(\mathsf{pk}_S, \mathsf{pk}_V, m, \sigma)$, given (m, r, s) and pk_V:

1. Compute $\widetilde{G}$ and R as in Section 4.1; the witness is $w = s^{-1} \bmod q$.
2. Pick $k \xleftarrow{\$} \mathbb{Z}_q$, set $A_L := k \cdot \widetilde{G}$. (real, left)
3. Compute $c_R := \mathsf{H}_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_L, m)$. (challenge, right)
4. Pick $z_R \xleftarrow{\$} \mathbb{Z}_q$, set $A_R := z_R \cdot G - c_R \cdot \mathsf{pk}_V$. (sim., right)
5. Compute $c_L := \mathsf{H}'_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_R, m)$. (challenge, left)
6. Compute $z_L := k + c_L \cdot w \bmod q$. (real, left)

The DV-signature is $\widehat{\sigma}_{\mathrm{AOS}} := (R, c_R, z_L, z_R)$.

Verification and simulation. $\mathsf{DVerify}$ recomputes $A_L := z_L \cdot \widetilde{G} - c_L \cdot R$ and $A_R := z_R \cdot G - c_R \cdot \mathsf{pk}_V$, then checks $c_R = \mathsf{H}_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_L, m)$ and $c_L = \mathsf{H}'_{\mathrm{FS}}(R, \widetilde{G}, \mathsf{pk}_V, A_R, m)$. The simulator Sim uses the analogous trick: pick $R \xleftarrow{\$} \langle G \rangle$, compute $\widetilde{G}$ as in honest verification, run the right clause with the real witness sk_V and simulate the left.

Relationship to Tso et al. [27] and trade-off. The construction above is the ECDSA-DL case of [27], simplified using $R = s^{-1}\widetilde{G}$ rather than the auxiliary $h = s/t$. Tso et al.'s h plays the role of our s^{-1} up to inversion, and their ζ is our R; the security analysis can be adapted with only minor modifications. The AOS variant yields $\widehat{\sigma}_{\mathrm{AOS}}$ with one group element and three scalars (vs. three group elements and three scalars for CDS), saving roughly 64 bytes on P-256. The cost is conceptual: two sequential Fiat–Shamir hashes and a security argument routed through AOS OR-proof analysis [1] rather than the textbook CDS extractor. For an EUDI-Wallet deployment, where the DV-signature is one component of a multi-kilobyte SD-JWT, we view CDS as the more deployable choice; AOS is the right choice where every byte matters (e.g., NFC-bound presentations).

4.3 Scope: The Single-Show Model

The construction of Section 4.1 is analysed in the single-show setting: each ECDSA-signed credential is presented to at most one Relying Party. We discuss why this assumption is appropriate for the EUDI-Wallet, how it interacts with the holder-binding signature, what would go wrong if it were dropped, and how multi-show deniability could in principle be added.

Why single-show fits the EUDI deployment. Single-show is not an additional restriction of our construction; it already reflects the EUDI-Wallet deployment model for obtaining Relying-Party unlinkability without anonymous credentials [15,8]. Reusing the same ECDSA-signed credential exposes the same issuer signature and is therefore trivially linkable. The current mitigation is *batch issuance* [2,8]: the issuer provides a fresh ECDSA signature for each anticipated presentation. Our construction then additionally gives per-presentation deniability by designating each credential to a single verifier using the protocol of Section 4.1 and the resulting DV-signature is non-transferable by Theorem 1.

The holder-binding signature. SD-JWT presentations include a Key Binding JWT (KB-JWT) [18]: an ECDSA signature by a holder-controlled binding key sk_H over the verifier nonce, a timestamp, and a hash of the disclosed credential. The KB-JWT is a publicly verifiable, transferable signature. However, it does not undermine the deniability of Theorem 1, because pk_H is an anonymous public key: a leaked KB-JWT – even though it contains a hash of the disclosed credential – only shows that *some* anonymous key pk_H authenticated to a verifier; without a transferable issuer signature, neither the credential hash nor the disclosed attributes themselves constitute evidence of issuance, and the KB-JWT therefore adds no transferable link to a real identity.

This argument again relies on the single-show assumption. Reusing pk_H across presentations would make them linkable, and later identification of pk_H would retroactively attribute the full set. In the current EUDI-Wallet model – where each credential uses a fresh binding key and is shown at most once [8] – this does not arise. Extending to multi-show would require either rotating pk_H per presentation (not supported by secure-element-bound keys) or applying the OR-proof technique to the KB-JWT.

What goes wrong with multi-show. If the same ECDSA-signed credential is presented to two verifiers V_1 and V_2, both DV-signatures reveal the same point R, since $R = kG$ is determined by the per-signature randomness of the underlying ECDSA signature, not by the presentation. A third party obtaining both transcripts (e.g., through breaches at both verifiers or an aggregator collecting leaked presentation data) can match the two R values and conclude that they originate from the same credential. Each transcript remains non-transferable in the sense of Definition 4 – either V_1 or V_2 could have produced its own transcript – but the *correlation* between transcripts is not simulatable by either verifier alone. The standard single-transcript deniability definition therefore does not capture

this joint property. This particularly matters under leakage from non-colluding verifiers [5]; colluding verifiers could trivially simulate correlated transcripts by coordinating their choice of R.

Multi-show deniability would require hiding R across presentations, e.g., by committing to R via a Pedersen commitment and proving the ECDSA verification relation on the committed value. This introduces the difficulty of handling $x_R \bmod q$, which will require more expensive techniques. Another approach is to combine designated-verifier techniques with an anonymous credential scheme, adding the OR-clause for sk_V to the presentation proof; this yields a clean composition providing both multi-show unlinkability and deniability [12].

5 Conclusion

This paper addressed the question whether ECDSA-signed credentials, as adopted by the EUDI Wallet, inherently lack plausible deniability compared to the authenticated channel approach of the German eID. We have shown that this perceived trade-off is a false dichotomy: the absence of deniability is not an intrinsic property of signature-based credentials, but of how they are presented.

Our core contribution is a lightweight UDVS construction that composes a Schnorr proof of knowledge of the ECDSA witness s^{-1} with a proof of knowledge of the verifier's secret key via the CDS OR-technique. The resulting designated-verifier signature achieves unconditional non-transferability while requiring no changes to the signature scheme, credential format, or issuer-side PKI. The construction integrates naturally into the single-show model already assumed for Relying-Party unlinkability, and its overhead – three group elements and three scalars (CDS) or one group element and three scalars (AOS) – is negligible relative to a multi-kilobyte SD-JWT presentation.

Two caveats frame the scope of our result. First, cryptographic deniability is necessary but not sufficient for real-world deniability: operational retention practices, audit logs, and contextual evidence can provide transferable conviction independently of the transcript. The cryptographic property is a structural prerequisite – without it, the transcript itself constitutes transferable evidence that no operational mitigation can retroactively remove – but it must be complemented by architectural and legal safeguards. Second, extending deniability to the multi-show setting remains an interesting open problem, requiring to hide the correlatable point R across presentations.

Acknowledgments. The authors used AI tools (Claude/Anthropic) to assist with drafting and improving the presentation of the manuscript. All AI-generated output was reviewed, edited, and verified by the authors, who take full responsibility for the final content.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Advances in Cryptology — ASIACRYPT 2002. Springer Berlin Heidelberg (2002)

2. Baum, C., Blazy, O., Camenisch, J., Hoepman, J.H., Lee, E., Lehmann, A., Lysyanskaya, A., Mayrhofer, R., Montgomery, H., Nguyen, N.K., Preneel, B., abhi shelat, Slamanig, D., Tessaro, S., Thomsen, S.E., Troncoso, C.: Cryptographers' feedback on the EU digital identity's ARF (2024)
3. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. Association for Computing Machinery (2006)
4. Bertram, M., Richter, M., Margraf, M.: Achieving third-party deniability in signature-based credential systems. Cryptography (2024)
5. Bertram, M., Zengin, B., Buchmann, N., Margraf, M.: Strong non-transferability from randomizable universal designated verifier signatures. Cryptography (2026)
6. Bundesamt für Sicherheit in der Informationstechnik (BSI): Technical guideline TR-03127: Architecture electronic identity card and electronic resident permit (2011), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03127/BSI-TR-03127_en.pdf?__blob=publicationFile&v=1
7. Bundesamt für Sicherheit in der Informationstechnik (BSI): Technical guideline TR-03110-2: Advanced security mechanisms for machine readable travel documents and eIDAS token (2016), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-2-V2_2.pdf?__blob=publicationFile&v=1
8. Bundesministerium für Digitales und Staatsmodernisierung: Blueprint for the EUDI Wallet Ecosystem in Germany (2023), https://bmi.usercontent.opencode.de/eudi-wallet/eidas-2.0-architekturkonzept/
9. Bundesministerium für Digitales und Staatsmodernisierung: Blueprint for the EUDI Wallet Ecosystem in Germany – Wallet Function: PID Issuance and Presentation (2023), https://bmi.usercontent.opencode.de/eudi-wallet/eidas-2.0-architekturkonzept/content/archive/00-pid-issuance-and-presentation/
10. Collins, D., Colombo, S., Huguenin-Dumittan, L.: Real-world deniability in messaging. Proceedings on Privacy Enhancing Technologies (2025)
11. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in Cryptology — CRYPTO '94. Springer Berlin Heidelberg (1994)
12. Debes, H.B., Giannetsos, T.: Retract: Expressive designated verifier anonymous credentials. In: Proceedings of the 18th International Conference on Availability, Reliability and Security. ACM (2023), https://doi.org/10.1145/3600160.3600191
13. Deutscher Bundestag: Ausschuss für Digitales: Stellungnahme zur Anhörung Digitale Identitäten, https://www.bundestag.de/resource/blob/901724/67441ef0b9ead7f9f09f15cc524226ad/Fluepke-Carl-Fabian-Luepke-.pdf
14. epicenter.works: Data Protection Analysis: eIDAS Architecture Reference Framework 1.4, https://epicenter.works/fileadmin/medienspiegel/user_upload/epicenter.works_-_ARF1.4.pdf
15. European Commission: The European Digital Identity Wallet – Architecture and Reference Framework (ARF) (2025), https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework, version 2.9.0
16. European Parliament and Council of the European Union: Directive (eu) 2015/849 (amld) (2015), https://eur-lex.europa.eu/eli/dir/2015/849/oj
17. European Parliament and Council of the European Union: Regulation (EU) 2024/1183 amending regulation (EU) no 910/2014 (eIDAS 2.0) (2024), https://eur-lex.europa.eu/eli/reg/2024/1183/oj

18. Fett, D., Yasuda, K., Campbell, B.: Selective disclosure for JWTs (SD-JWT). IETF Internet-Draft, draft-ietf-oauth-selective-disclosure-jwt (2025)
19. Frigo, M., abhi shelat: Anonymous credentials from ECDSA (2024), https://eprint.iacr.org/2024/2010
20. Hao, F.: Rfc 8235: Schnorr non-interactive zero-knowledge proof (2017), https://www.rfc-editor.org/rfc/rfc8235
21. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Advances in Cryptology — EUROCRYPT '96. Springer Berlin Heidelberg (1996)
22. Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The BBS signature scheme (2024), https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/
23. Monnerat, J., Pasini, S., Vaudenay, S.: Efficient Deniable Authentication for Signatures: Application to Machine-Readable Travel Document. In: Applied Cryptography and Network Security. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
24. Rastegari, P., Berenjkoub, M., Dakhilalian, M., Susilo, W.: Universal designated verifier signature scheme with non-delegatability in the standard model. Information Sciences (2019)
25. Reitinger, N., Malkin, N., Akgul, O., Mazurek, M.L., Miers, I.: Is cryptographic deniability sufficient? Non-Expert perceptions of deniability in secure messaging. In: 2023 IEEE Symposium on Security and Privacy (SP) (2023)
26. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Advances in Cryptology - ASIACRYPT 2003. Springer Berlin Heidelberg (2003)
27. Tso, R., Gonzàlez Nieto, J.M., Okamoto, T., Boyd, C., Okamoto, E.: Verifier-key-flexible universal designated-verifier signatures. In: Cryptography and Coding. Springer Berlin Heidelberg (2007)
28. Vaudenay, S.: Deniable RSA Signature: The Raise and Fall of Ali Baba. In: Cryptography and Security: From Theory to Applications. Springer Berlin Heidelberg (2012)
29. Verbraucherzentrale Bundesverband (vzbv): Report on the architecture of the EUDI-wallet (2024), https://www.vzbv.de/sites/default/files/2025-02/vzbv-Gutachten%20zur%20Architektur%20der%20EUDI-Wallet.pdf
30. Viola Heeger: EUDI-Wallets: Streit um signierte Daten, https://background.tagesspiegel.de/digitalisierung-und-ki/briefing/eudi-wallets-streit-um-signierte-daten
31. Yadav, T.K., Gosain, D., Seamons, K.: Cryptographic deniability: A multi-perspective study of user perceptions and expectations. In: 32nd USENIX Security Symposium (USENIX Security 23). USENIX Association (2023)