EUDI Wallet & Linux Phones

What it is, how it works, and why your Mobile Linux can’t use it yet

Benedikt Wildenhain
benedikt.wildenhain@hs-bochum.de
https://ruhr.social/@benedikt
https://gitlab-ce.hs-bochum.de/hardwarenahe-it/eudi-wallet/

2026-05-23

What is the EUDI Wallet?

EU Digital Identity Wallet — a state-issued digital identity infrastructure mandated by eIDAS 2.0 (Regulation (EU) 2024/1183, in force May 2024)

- Every EU Member State must offer at least one wallet to all citizens and residents
- **Deadline: 20 November 2026**
- Must achieve **Level of Assurance: High** (LoA High) — same bar as a passport

What it holds:

- **PID** — Person Identification Data (name, DOB, nationality, …)
- **(Q)EAA** — (Qualified) Electronic Attestations of Attributes (driving licence, diplomas, prescriptions, …)

What it Replaces / Complements

- eID cards for online authentication
- Physical document presentation at counters and borders
- Fragmented national identity silos

Why it matters for Linux users: The wallet will become the primary digital identity mechanism for EU citizens interacting with public services, banks, healthcare, and more. If you can't run it, you are effectively excluded from digital public life.

The Actor Landscape

Actor	Role	Germany
Member State / Wallet Provider	Distributes app, issues WIAs	BMI ↗ + SPRIND ↗
PID Provider	Issues identity credential into wallet	BSI / BMI
Attestation Providers	Issue driving licences, diplomas, …	Any accredited body
Relying Parties	Verify presentations from the wallet	Public authorities, private sector
QTSP	Operates remote HSM holding private keys	TBD

The citizen holds the wallet. The citizen controls what is disclosed. The citizen does **not** control the software stack.

The Trust Chain (1/2): Device Check and WIA

Step 1 — Device check (MDVM)

The Wallet Provider Backend checks whether your device is “sufficiently secure.” Primary signal: **Android Key Attestation** — a hardware-rooted certificate chain terminating at a Google-provisioned root CA embedded at device manufacture. (BMI MDVM architecture↗)

Step 2 — Wallet Instance Attestation (WIA)

If the device passes, the Wallet Provider issues a **WIA**: a signed credential certifying *“this specific app binary on this specific device is genuine.”*

> *The WIA is the gateway credential. Everything downstream depends on it. Without a WIA, no PID Provider will talk to you.*

The Trust Chain (2/2): PID Issuance and Presentation

Step 3 — PID issuance

The PID Provider validates the WIA. If valid, it issues your identity credential into the wallet.

Step 4 — Presentation

You present attributes to a Relying Party using **OpenID4VP** (HTTP-based). Selective disclosure: you reveal only what is asked for.

Key point for Linux users: Steps 1 and 2 are where Linux phones are excluded. Steps 3 and 4 use open, HTTP-based protocols that are platform-agnostic in principle.

The German Implementation: Status

Architecture version: v0.9.1 (current, May 2026)

Source code: not yet published — promised “later in 2026”

App platforms: Android and iOS only (initial launch)

Wallet Provider: BMI / SPRIND — single provider at launch

Key tension: The architecture is publicly documented and uses open protocols, but the app binary is closed, obfuscated, and the only path to a WIA.

The German Implementation: Design Choices

Component	Choice	Status
Primary device-integrity signal	Android Key Attestation	Confirmed, load-bearing
Secondary signal	Play Integrity API	Optional — may be dropped
Key storage	Local TEE/SE or remote QTSP HSM	Both supported
App integrity	RASP + code obfuscation	Load-bearing
Reproducible builds	Not possible	Explicitly ruled out

BMI statement (issue #9, 2026-05-11): *"Should the evaluation conclude that Play Integrity does not offer any significant additional security benefit, it will not form part of the final architecture."*

The Cryptographic Key Architecture: Local WSCD (Wallet Secure Cryptographic Device)

Keys stored in the device's TEE (Trusted Execution Environment) or Secure Element:

Step	Component
1	Device TEE/SE generates key pair
2	Android Keystore (HARDWARE / STRONGBOX) wraps it
3	Key Attestation cert chain rooted at Google root CA
4	MDVM check passes -> WIA issued by Wallet Provider

This is the default path. Requires a device with a Google-provisioned TEE root. See https://eudi.dev/latest/architecture-and-reference-framework-main/#43-reference-architecture

The Cryptographic Key Architecture: Remote WSCD

Keys stored on a QTSP-operated HSM (Remote WSCD / RWSCD):

Step	Component
1	QTSP server holds keys (certified HSM, EN 419221-5)
2	User authenticates via PIN to QTSP
3	QTSP signs on behalf of user
4	Device **still** needs to pass MDVM for WIA

*Even with RWSCD, the **device** must currently pass MDVM. The keys are off-device, but the device-integrity gate remains. This is a policy choice — not an architectural necessity.*

Why Your Linux Phone Can’t Use It: Blocker 1

Blocker 1 — WIA gate *(hardest; structural; eIDAS 2.0 by design)*

The Wallet Provider only issues WIAs to the **official BMI-signed binary**. A sideloaded APK — regardless of runtime environment — gets no WIA.

- This is not a Linux-specific problem: it also stops GrapheneOS users
- It is the intended design: the Wallet Provider vouches for the app instance
- Removing it would require BMI to issue WIAs to self-built binaries (they won’t) or a separately certified Wallet Provider trusted by PID Providers

Consequence: even a perfect Android emulation layer on Linux is blocked here unless BMI explicitly supports it.

Why Your Linux Phone Can’t Use It: Blockers 2 and 3

Blocker 2 — Android Key Attestation *(hard; platform-level)*

Key Attestation requires a hardware-rooted cert chain from the device’s TEE, provisioned by Google at manufacture. Linux phones have no such chain. Neither Waydroid nor ATL can provide hardware-backed Key Attestation. (See BMI issue #2↗)

Blocker 3 — Play Integrity *(medium; may disappear)*

Requires Google Play Services and a live call to Google’s servers. Currently **optional** — BMI is evaluating whether to include it at all. (BMI issue #9↗)

- If dropped: this blocker disappears entirely
- If retained: Waydroid + GAPPS can satisfy it; ATL cannot

Waydroid: What It Is

Waydroid — Android 13 (LineageOS-based) in a Linux namespace container.

- Direct hardware passthrough (GPU, camera, sensors)
- Works on ARM Linux phones: PinePhone Pro, Librem 5, and others
- Optional GAPPS image includes Google Play Services
- Play certification: manual self-registration at google.com/android/uncertified, requires Google account

What it can do relevant to the wallet:

- Install and launch the official BMI APK (once published)
- Satisfy Play Integrity via GAPPS + manual Play certification
- Run on mainline Linux phones today

Waydroid: Why It Still Fails

The specific failure: software-backed Android Keystore

The Android Keystore inside Waydroid is **software-backed** (`SOFTWARE` security level, not `HARDWARE` or `STRONGBOX`).

Key Attestation certificates from a software keystore will not satisfy the MDVM's LoA High requirement — the cert chain cannot be rooted in a Google-provisioned hardware key without TEE passthrough.

Theoretical fix: A TEE/Keymaster HAL bridge forwarding Key Attestation requests from the Waydroid container to the host device's physical TEE. This engineering work **does not exist today.**

ATL: What It Is

Android Translation Layer — reimplements Android APIs natively on Linux. Think Wine, but for Android. No full Android system running in parallel.

- EU-funded: NLnet / NGI Mobifree, grant started August 2024
- Lighter weight than Waydroid, faster startup, better desktop integration
- Apps run as first-class Linux processes — native notifications, no container overhead
- Aligned with the Linux phone ecosystem's long-term goals

Current status: Early development.

ATL: Why It Can’t Run the EUDI Wallet

Fundamental gaps for security-critical apps:

- No Android Keystore HAL — no hardware attestation path of any kind
- No Google Play Services — Play Integrity impossible
- No RASP/obfuscation compatibility tested or expected
- The known-working app list demonstrates the gap: simple games with no security requirements; a RASP-hardened identity app is far out of scope

However: ATL is the architecturally correct long-term direction. If it matures to support hardware-adjacent APIs and gains a Keystore HAL, the picture changes significantly.

ATL is the right long-term direction. It is not a 2026 solution.

Paths Forward: Overview

Approach	Works today?	Key blocker
Waydroid + official APK	No	Key Attestation software-only
ATL + official APK	No	No Keystore HAL; early-stage
Native Linux wallet app	No	Does not exist
RWSCD + relaxed MDVM policy	No (policy)	BMI policy, not architecture
Web/browser wallet	No	Does not exist for Germany
GrapheneOS + Sandboxed Play	Possibly	Play Integrity optional

Two paths are **tractable without new engineering** — they require BMI to act.

Paths Forward: The Two Tractable Options

Option 1 — RWSCD + relaxed MDVM policy

Keys are already off-device in the RWSCD path. If the device's TEE is not holding any key material, the security argument for requiring hardware Key Attestation weakens considerably. This is a **policy decision by BMI**, not an engineering problem.

Option 2 — Web/browser wallet

eIDAS 2.0 does not mandate a mobile app. OpenID4VP and OpenID4VCI are HTTP-based protocols. A browser flow + RWSCD + PIN authentication is architecturally compliant and fully platform-agnostic. Denmark's MitID chip does exactly this today.

Neither option exists for Germany yet. Both are architecturally sound.

The Structural Constraint: eIDAS 2.0 Itself

The Wallet Provider dependency is **not a BMI design choice** — it is the legal architecture:

- eIDAS 2.0 Art. 5a: Member States must *provide* the wallet
- The Wallet Provider issues the WIA — the trust anchor for the entire ecosystem
- A citizen cannot opt out of the Wallet Provider, just as they cannot opt out of the issuing state for a passport

What the wallet gives citizens:

- **(yes) Data sovereignty** — selective disclosure, citizen controls what is revealed
- **(yes) Key sovereignty** — private keys in TEE or QTSP HSM, not accessible to Wallet Provider
- **(yes) Revocation control** — citizen can revoke their own WIA
- **(no) Software sovereignty** — citizen does not control the software stack

The wallet is state-issued credential infrastructure, not a self-sovereign identity system.

What To Watch: Open Issues

BMI GitLab

- **Issue #9 — Play Integrity:** Team confirmed optional, may be dropped entirely. Follow for the final architecture decision before November 2026.
- **Issue #10 — Non-mobile users:** MitID-style hardware key / web wallet proposal. No team response after 2 months. **Community voices needed** — the team is responsive to technical arguments.

The RWSCD + relaxed MDVM path is a **policy ask, not an engineering ask**. It is the shortest path to Linux phone support and can be argued on the team's own architectural terms.

What To Watch: Milestones

Event	Expected	Significance for Linux
BMI source code publication	“Later 2026”	Audit MDVM; assess Waydroid feasibility
QTSP partner named	Unknown	RWSCD architecture becomes concrete
Play Integrity decision	Before Nov 2026	Removes one blocker if dropped
ATL Keystore HAL support	Unknown	Long-term path opens
BMI wallet launch	Nov 2026	eIDAS 2.0 deadline

Summary

Layer	Situation
eIDAS 2.0 mandates LoA High	Hardware attestation required
Hardware attestation	Android/iOS only today
WIA gate	Official binary only — no self-builds, no alternative runtimes
Linux phones	No hardware Key Attestation available
Waydroid	Closest option — blocked by software-only keystore
ATL	Right direction — years away from viability
Path 1	RWSCD + BMI relaxes MDVM policy (policy ask)
Path 2	Web/browser wallet (architecture ask)

Summary — Takeaway

The protocols are open (OpenID4VP, OpenID4VCI, SD-JWT, mdoc). The architecture supports remote keys. The legal framework does not mandate a mobile app.

But: the blockers are policy and implementation choices, and those are no easier to shift than technical ones — they require political will, budget, and a willingness to broaden the platform scope.

For Linux phone users in the near term, the realistic expectation is **exclusion from the default wallet** and reliance on workarounds (GrapheneOS + Sandboxed Play, web flows where available, or a second device) until that political shift happens.

References

BMI wallet documentation

- Architecture v0.9.1: https://bmi.usercontent.opencode.de/eudi-wallet/wallet-development-documentation-public/v0.9.1/
- GitLab (issues): https://gitlab.opencode.de/bmi/eudi-wallet/wallet-development-documentation-public/
- Issue #2 (Key Attestation): https://gitlab.opencode.de/bmi/eudi-wallet/wallet-development-documentation-public/-/issues/2
- Issue #4 (Reproducible builds): https://gitlab.opencode.de/bmi/eudi-wallet/wallet-development-documentation-public/-/issues/4
- Issue #9 (Play Integrity): https://gitlab.opencode.de/bmi/eudi-wallet/wallet-development-documentation-public/-/issues/9
- Issue #10 (Non-mobile users): https://gitlab.opencode.de/bmi/eudi-wallet/wallet-development-documentation-public/-/issues/10

References (cont.)

Legal

- eIDAS 2.0 (Reg. (EU) 2024/1183): https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401183

Android-on-Linux projects

- Waydroid: https://waydro.id/
- ATL (Android Translation Layer): https://gitlab.com/android_translation_layer/android_translation_layer
- NLnet ATL grant: https://nlnet.nl/project/ATL/

Comparable implementations

- Denmark MitID chip: https://www.mitid.dk/en-gb/get-started-with-mitid/how-to-use-mitid/mitid-chip/

Discussion

Questions? Comments? Corrections?

Participate Hackathon 2026-06-04/05?
https://eudi-wallet.gov.de/news/eudi-wallet-hackathon-2026

Email	benedikt.wildenhain@hs-bochum.de
Mastodon	https://ruhr.social/@benedikt

Slides and sources:

https://gitlab-ce.hs-bochum.de/hardwarenahe-it/eudi-wallet/

 Documents gathering and some reading by hand, summarizing unproudly done with assistance from Anthropic Claude Sonnet 4.6 / Opus 4.6 due to time constraints.