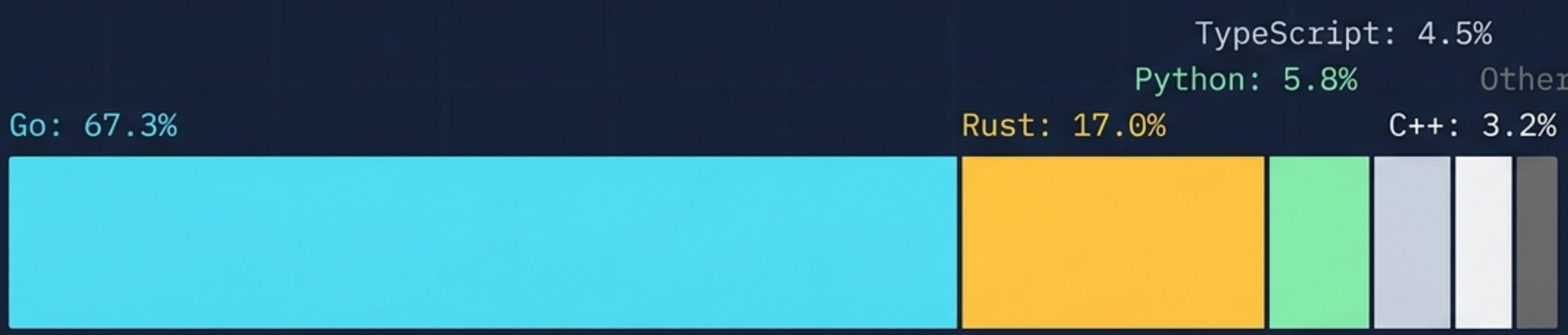
prldroid-wallet: Native Android Pearl Wallet

Bridging Jetpack Compose and Go for a no-compromise mobile blockchain experience.

Author : Robin L. M. Cheung, MBA
Repository : git.robin.mba/rcheung/prldroid-wallet
Platform : Forgejo (Beyond coding. We Forge.)

The Repository at a Glance



Size
84 MiB

Structure
2 branches, 20 commits

A multi-language mobile powerhouse. The repository embeds the actual Pearl AI monorepo to power its core, leveraging heavy Go and Rust implementations right on the device.

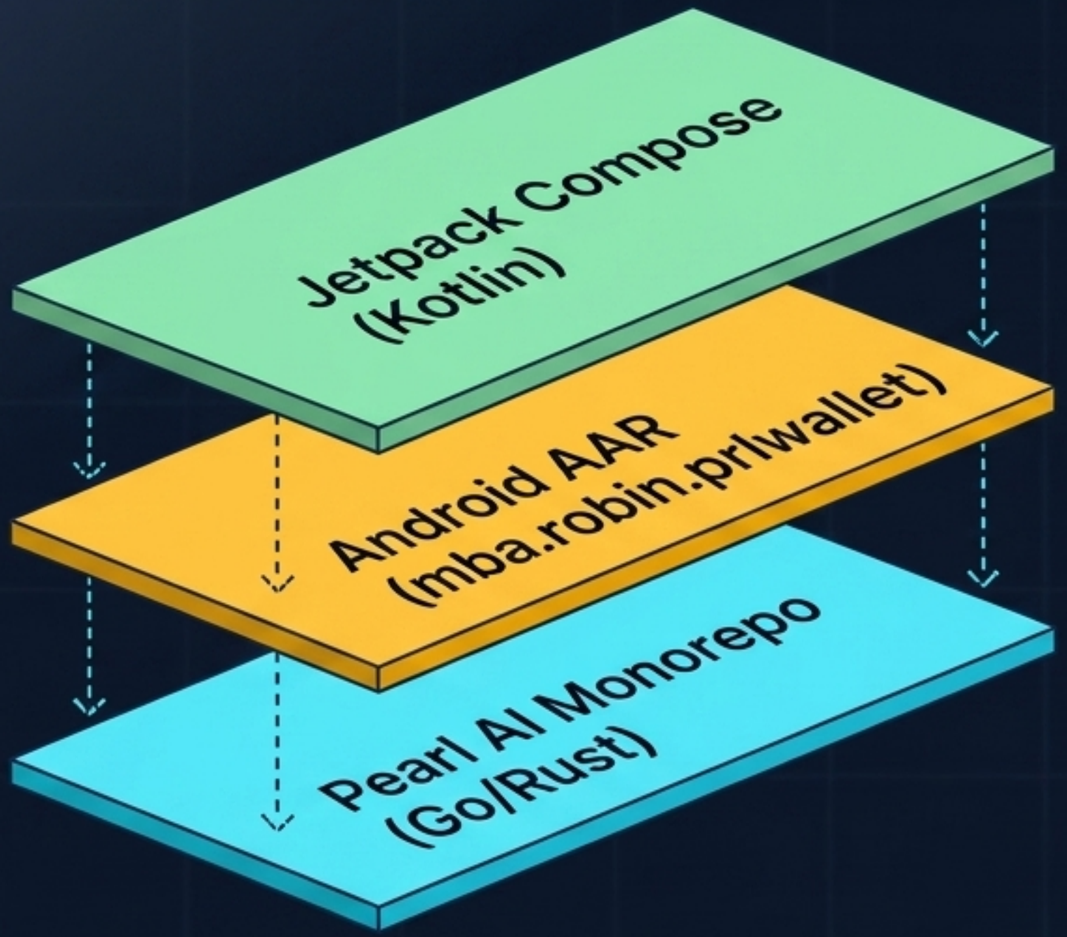
Native Android UX: The Glassmorphic Interface

A multi-paradigm interface combining glassmorphic elements with minimalist data visualization. The UX prioritizes clean lines, high-contrast data presentation, and intuitive navigation for native Android implementation, ensuring clarity and efficiency for power users.

The No Reimplementation Mandate

Approach 1: WebView / HTML5	Drawback: Poor performance, compromised UX, relies on external signing servers.
Approach 2: React Native / Cross-Platform	Drawback: Requires rewriting complex cryptography and UTXO selection logic in JavaScript/TypeScript. High risk of bugs.
✓ Approach 3: Jetpack Compose + gomobile bind (prldroid)	Advantage: **Zero reimplementation.** Compiles the existing, battle-tested Go wallet directly to an **Android AAR. The exact same debugged Go code powering the Oyster daemon handles BIP-32/44/86 HD key derivation and transaction signing natively.**

The Architecture Stack

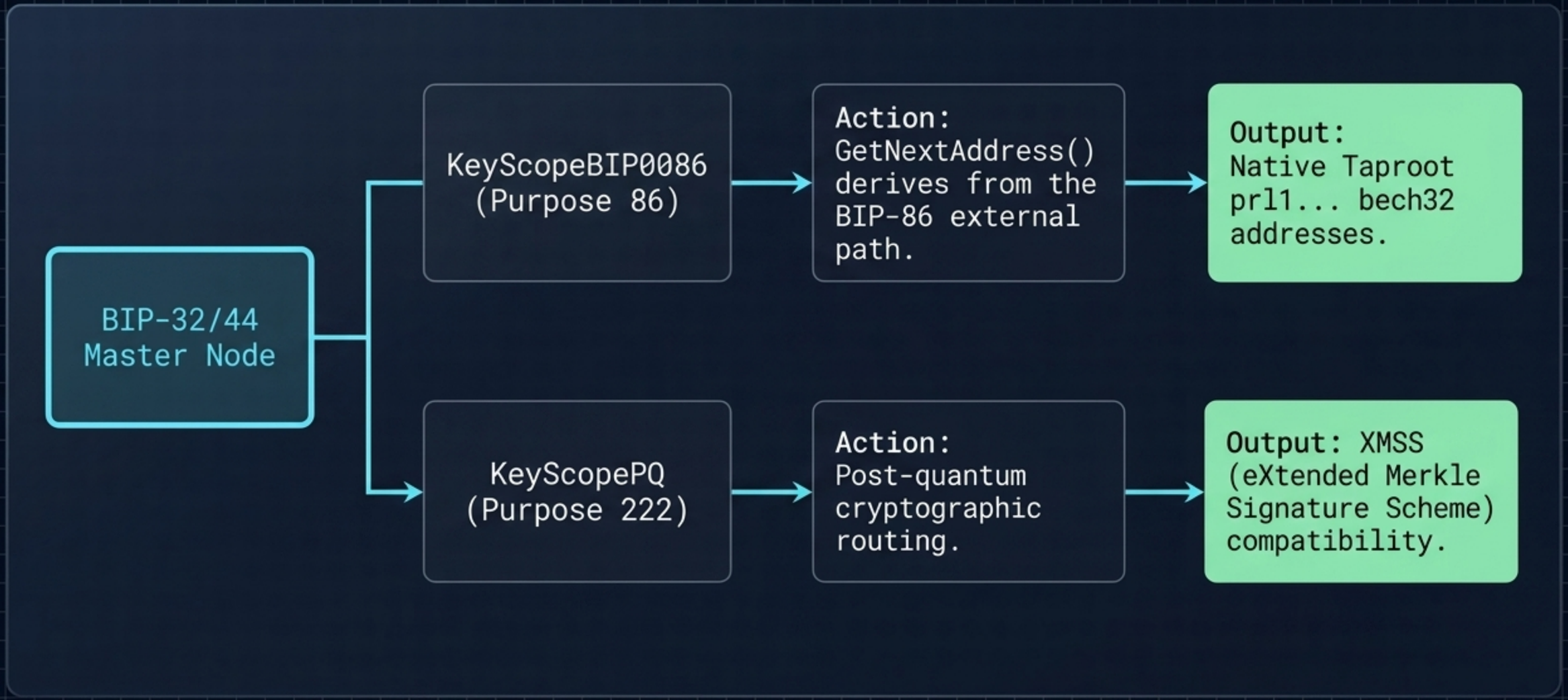


Pure native Android UI. Handles all theming, animations, and user interactions. No web-rendering overhead.

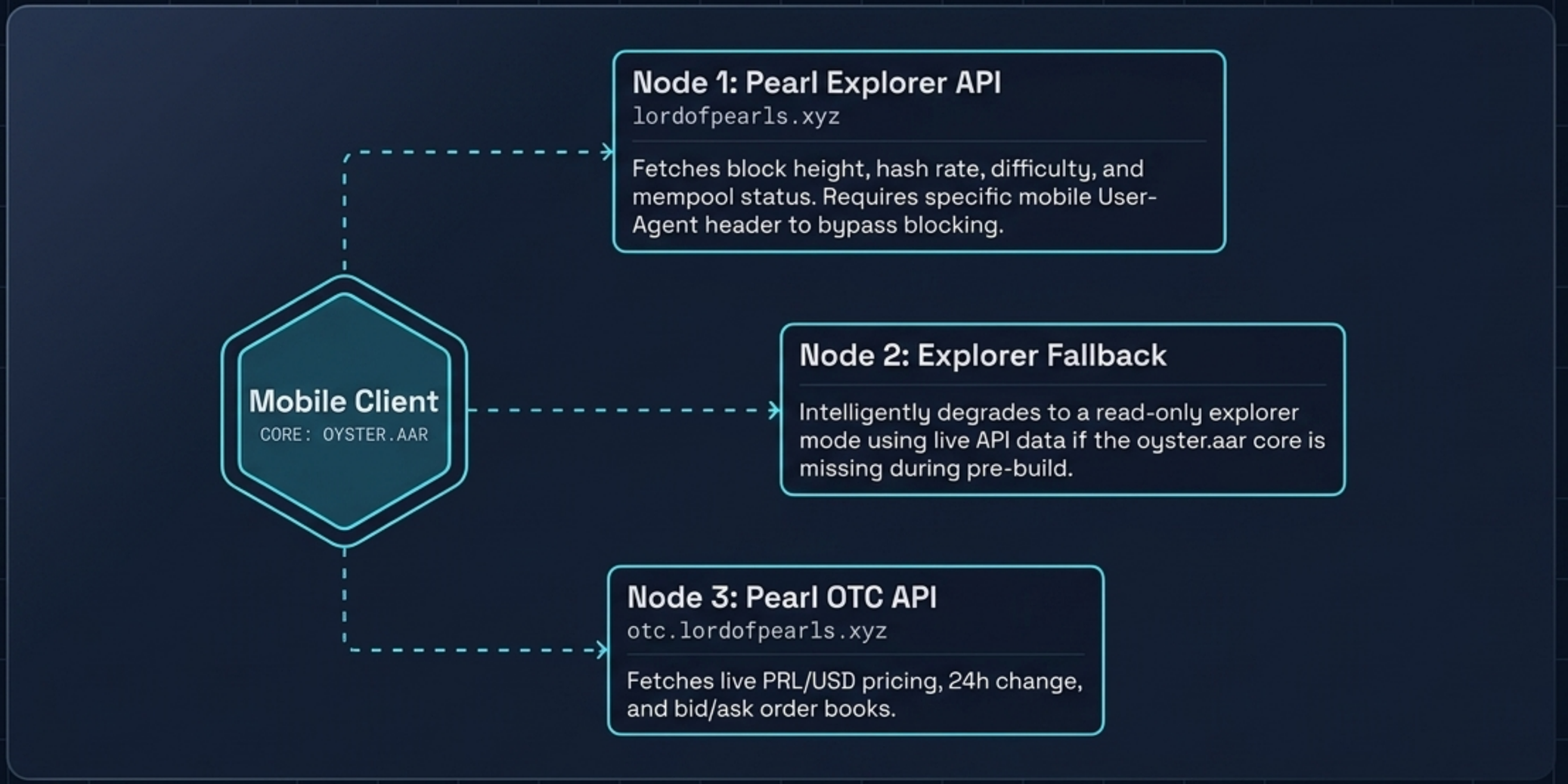
The compiled binary acting as the translation layer. Uses the **androidbridge** package to expose real monorepo functions without stubbing.

The embedded core. Runs the Oyster daemon's internals directly on the mobile CPU.

Cryptography & Key Derivation



Data Topology & The No Mocks Rule



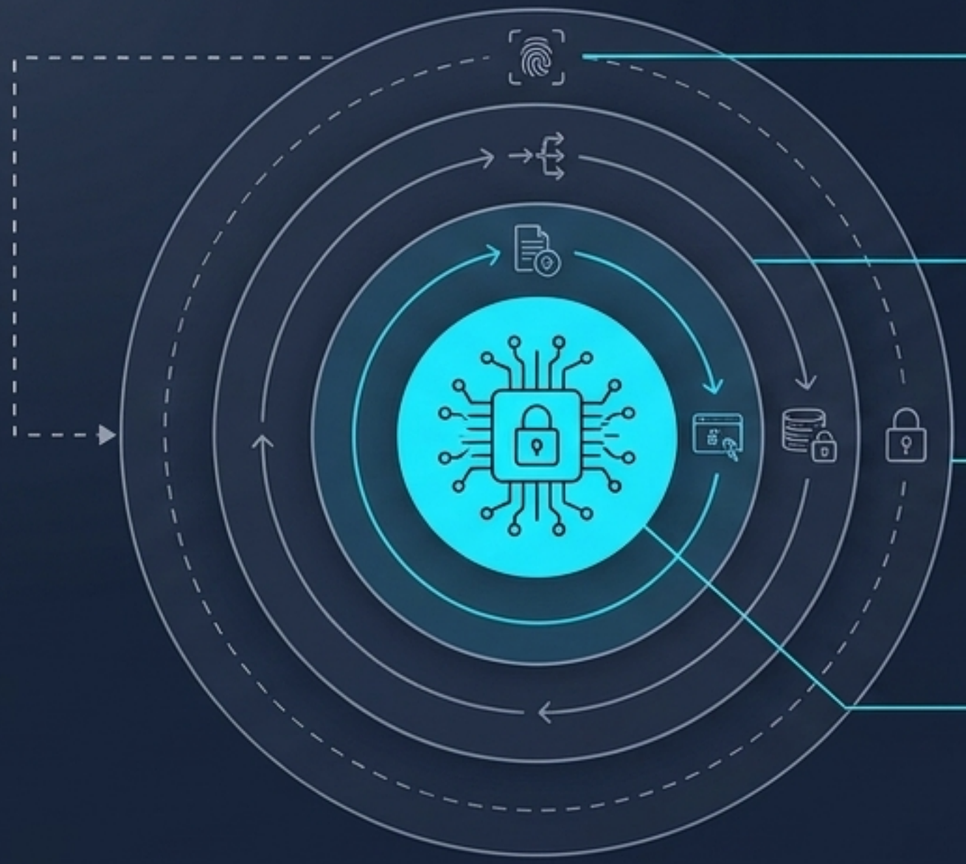
MARKET NOTE: The OTC platform charges a strict 1.8% fee on trades, settling PRL ↔ USDC instantly on-ledger.

Node Connectivity Matrix

The app reads BuildConfig.WALLET_ADDRESS as the default display. To connect the embedded Oyster wallet to a live pearld node, RPC ports must be precisely configured.

Network	pearld RPC Port	Oyster Legacy RPC Port
Mainnet	44107	44207
Testnet	44109	44209
Testnet2	44111	44211
Simnet	18556	18554

Security Posture: The Defense-in-Depth Vault



Ring 1 (Access)
Biometric gates (FaceID/Fingerprint) protecting passphrase unlock on the Transact screen.

Ring 2 (Hashing)
Scrypt key stretching via the snacl library.

Ring 3 (Encryption)
AES-256 encryption. Private keys are never stored unencrypted.

Core: BoltDB
`(Context.filesDir/oyster/wallet/wallet.db)`

⚠ Strict Blocker: 🚫
Cleartext traffic is blocked. Enforced TLS via network_security_config.xml

⚠ Strict Blocker: 🚫
Cloud backups strictly blocked via backup_rules.xml (excluding the wallet.db file).

Build Prerequisites: The Dual Environment

Go Engine Build

Go:
Version 1.21+

gomobile:
Installed automatically via `build-walet-aar.sh`

Android NDK:
Revision r25c+

Config:
`ANDROID_NDK_HOME` path strictly enforced.

Android App Build

IDE:
Android Studio Hedgehog+

Plugin:
Android Gradle Plugin 8.5.2

Language:
Kotlin 2.0.21

Targeting:
Compile SDK 35 / Min SDK 26 (Android 8.0)

The Compilation Pipeline

1. Source Acquisition

Clone monorepo into `wallet-go/`.

2. AAR Binding

```
gomobile bind
-target=android
-androidapi 26
-javapkg
mba.robin.prlwa
llet
```

3. Keystore Injection

Copy production keystore into `signing/`. Configure `keystore.proper ties`.

4. Typography Prep

Inject .ttf assets into `app/src/main/r es/font/`.

5. Gradle Assembly

Open in Android Studio, sync, and execute build.

Production & Release Management

.aab (App Bundle)

`outputs/bundle/release/app-release.aab`

↳ Routed to Play Console.

.apk

`outputs/apk/release/app-release.apk`

↳ Routed to Direct Distribution.

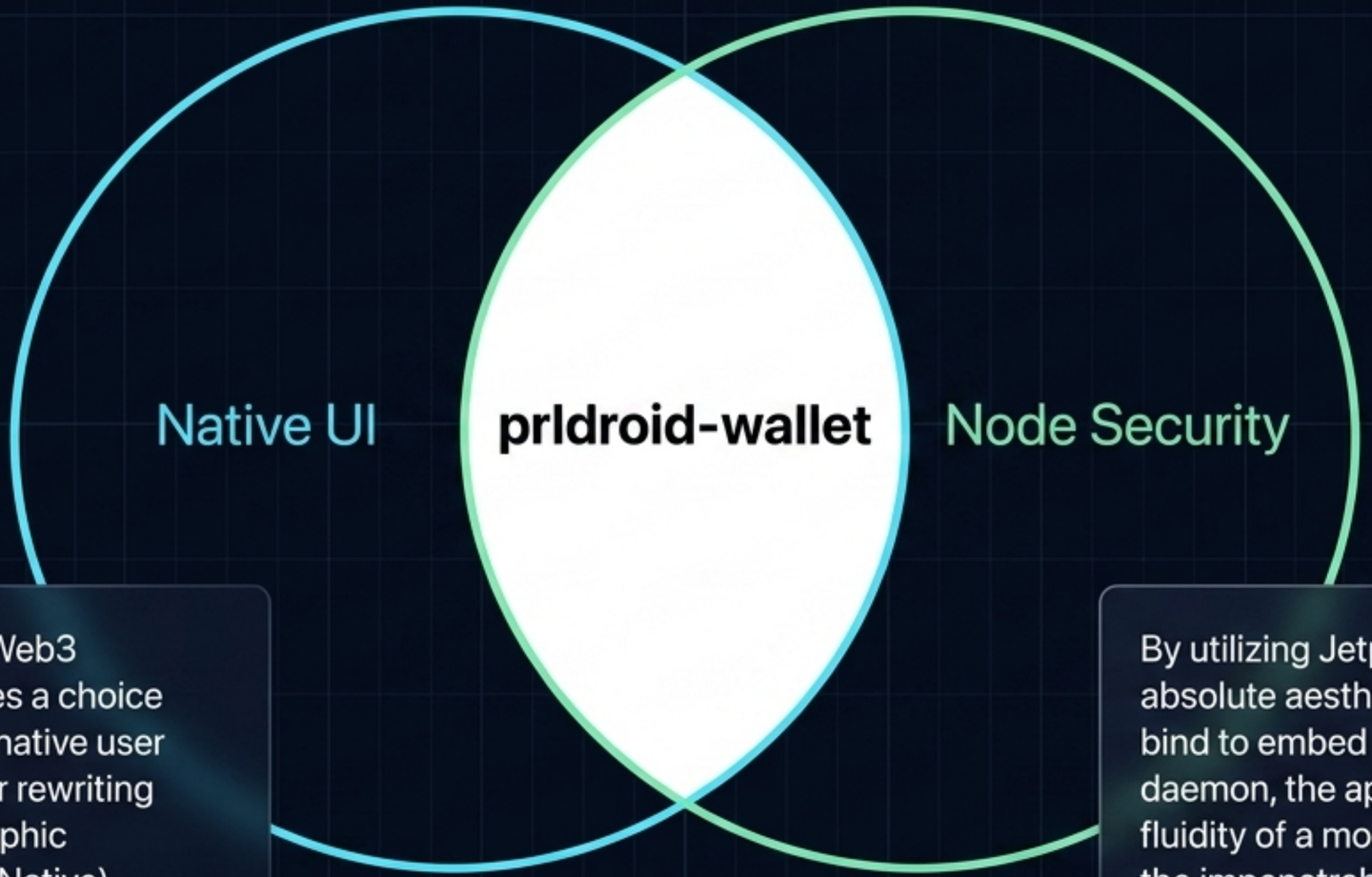
Native Debug Symbols

`native-debug-symbols.zip`

↳ Routed to Play Console for crash tracing.

⚠ **CRITICAL: signing/keystore.properties** and **signing/production.keystore** must remain in .gitignore. The HelloWord production keystore (May 2026, Alias: upload) fingerprint (56:C1:36...) is strictly protected. **Never commit** signing assets.

Synthesis: Form Meets Function Without Compromise



The standard tradeoff in Web3 mobile development forces a choice between a sluggish, non-native user experience (WebViews) or rewriting highly sensitive cryptographic logic from scratch (React Native).

By utilizing Jetpack Compose for absolute aesthetic control and gomobile bind to embed the unaltered Oyster Go daemon, the application achieves the fluidity of a modern Android app with the impenetrable security of a full blockchain node.

Project Information & Licensing

```
Licensing: Pearl node and wallet source: ISC License (wallet-go/LICENSE).
Android app code: © 2026 Robin L. M. Cheung, MBA. All rights reserved.

Platform Telemetry:
Powered by Forgejo Version 13.0.4.
Page render: 148ms | Template render: 11ms.
```

Beyond coding. We Forge.