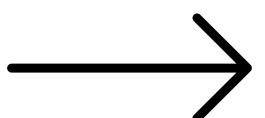


Bank-Grade Mobile Wallet Architecture: The Key to Wallet Core & Multi-Chain Mastery



Esmaeel Moradi

Senior Android Engineer



Bank-Grade Mobile Wallet Architecture:

The Key to Wallet Core and Multi-Chain Mastery

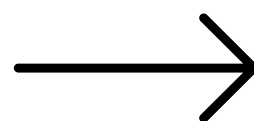
We know the heart of any mobile wallet is the **Wallet Core layer**.

A platform's success hinges on its ability to handle multiple, disparate protocols with a single cryptographic engine (Core), not by building a separate application for every single coin!



Esmaeetil Moradi

Senior Android Engineer



Defining Wallet Core and Security

The JNI Security Boundary: Protecting the Keys

The Core is responsible for **Key Derivation** based on standards (BIP-39/44/84) and **Transaction Signing**.

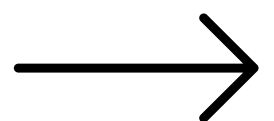
- **C++ (Native):** Sensitive logic is executed here to keep it isolated from the vulnerabilities of the Java/Kotlin environment.
- **C Interface (JNI):** A hard, secure layer that only allows specific functions (like `signTransaction`) to be called. The private key never leaves the Core.

Goal: To turn the Core into a secure "Black Box" that only accepts input and returns a signature, with zero key exposure.



Esmaeetil Moradi

Senior Android Engineer



Multi-Chain Protocol Challenges

Three Protocol Families:

The Challenge of Architectural Scale

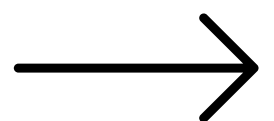
The Core must be built to "speak the language" of these protocols to be truly Multi-Chain:

- **UTXO Family (e.g., Bitcoin):**
 - **Model: Unspent Transaction Outputs (UTXO).** The Core must execute complex logic to select the correct UTXOs for transaction building.
- **EVM Family (e.g., Ethereum):**
 - **Model: Account-Based.** The Core must correctly handle Nonce management and Gas calculations within the transactions.
- **Non-EVM Family (e.g., Solana):**
 - **Challenge:** Requires the implementation of different cryptographic algorithms (like **Ed25519**) directly within the Core.



Esmaeail Moradi

Senior Android Engineer



The Adapter Model: The Bridge between Mobile and Core

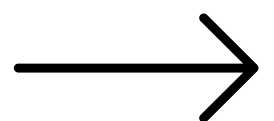
This is where the Adapter (Kotlin) layer acts as the crucial "translator."

1. **JSON Preparation:** The mobile application prepares the necessary JSON/Protobuf structure for the Core (e.g., for a Cosmos-SDK or EVM transaction).
2. **WalletConnect:** This protocol delivers the prepared JSON to the Wallet Core using specific methods (like `trust_signTransaction`).
3. **Adapter's Role:** This layer ensures that all protocol rules (like validating Fees or Gas) are adhered to before sending the request to the Core.



Esmaeeil Moradi

Senior Android Engineer



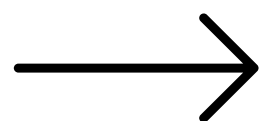
Final View: Architecting a Platform, Not Just an App

This modular, C++ Core-based architecture is the only way to ensure the **security and scalability** required for financial infrastructure. Trust Wallet Core demonstrates how a robust architecture can continuously embrace new blockchains.



Esmaeetil Moradi

Senior Android Engineer



? Technical Question for Experts:

For open-source projects with this level of complexity, could a tool that uses

**AI to automate documentation
visualization**

solve the code understanding bottleneck
faster than Core teams can?



Esmaeeil Moradi

Senior Android Engineer



System Design

From Code to Business Value

Architectural Insights

Let's Build a Scalable Future

by Esmaeel Moradi
Senior Android Engineer

Let's Connected

A series of concentric, curved yellow lines that sweep across the bottom right corner of the cover, creating a sense of motion and connectivity.