Software Requirements Specification (SRS) & Technical Architecture

Integration of Telebirr SuperApp SDK (In-App Payment)

Project Name: Suuq S - Multi-Vendor Marketplace
Document Version: 1.1
Date: February 6, 2026
Confidentiality Level: Confidential / INSA Audit Review Only
Prepared For: Information Network Security Administration (INSA), Ethiopia

---

1. Executive Summary

This document outlines the security architecture, data flow, and functional specifications for the integration of the Telebirr In-App Payment SDK within the "Suuq S" digital marketplace platform. It demonstrates compliance with financial security standards, focusing on the integrity of the **Merchant of Record** model where Suuq S acts as the primary payment acceptor before disbursing funds to sub-merchants (vendors).

2. System Architecture

2.1 Overview

The system operates on a Microservices-based architecture comprising three primary distinct entities:

1. **Client Layer (Flutter Mobile App):** Determines user intent and interfaces with the Telebirr Android/iOS SDK. Protected via ProGuard obfuscation (`com.huawei.ethiopia.pay.sdk.api.core.**`).
2. **Application Layer (NestJS Backend):** Acts as the secure orchestrator. It holds the RSA Private Keys, manages Fabric Tokens, and processes callbacks. Hosted at Static IP `134.209.94.162`.
3. **Gateway Layer (Ethio Telecom Fabric):** The external payment processor handling the ledger and fund movements.

2.2 Component Diagram

Interaction logic utilizes a "Server-Signed, Client-Executed" pattern to prevent client-side tampering.

- **Fabric Token Service:** A dedicated backend service manages the lifecycle of the Telebirr access token, auto-refreshing it before expiration to prevent transaction failures.
- **Crypto Module:** Handles RSA-2048 encryption and SHA256WithRSA signing.

---

3. Data Flow Diagram (DFD)

3.1 Transaction Lifecycle

The transaction follows a strict **Asynchronous Verification** model. The mobile app is never trusted with payment confirmation; only the server-to-server callback validates a purchase.

Sequence:

1. **Request Initialization:** User selects "Pay with Telebirr". Backend generates a signed `receiveCode` payload.
2. **SDK Handoff:** App invokes Telebirr App via SDK using the signed payload.
3. **Payment Execution:** User confirms PIN in Telebirr App.
4. **Callback (Notify URL):** Telebirr server POSTs result to `https://api.suuq.ugasfuad.com/api/payments/telebirr-callback`.
5. **Reconciliation & Disbursement:** Backend verifies signature, updates local ledger, and triggers 95/5 split via `/merchant/transfer`.

3.2 Mermaid.js Visualization

(Copy the code below into a Mermaid-compatible viewer or VS Code Preview)

```
sequenceDiagram
    participant User
    participant App as Suuq Flutter App
    participant Backend as NestJS Server (134.209.94.162)
    participant Telebirr as Telebirr Fabric System

    Note over Backend, Telebirr: Phase 1: Pre-Order (Initialization)
    User->>App: Checkout (Total: 100 ETB)
    App->>Backend: POST /api/v1/orders/initiate
    Backend->>Backend: Generate RSA-2048 Signature
    Backend->>Telebirr: Request Prepay ID (applyFabricToken)
    Telebirr-->>Backend: Return rawRequest & Sign
    Backend-->>App: Return Encrypted String (H5/SDK Payload)

    Note over App, Telebirr: Phase 2: Execution (Client Side)
    App->>Telebirr: Invoke SDK (startPay)
    User->>Telebirr: Enter PIN & Confirm
    Telebirr-->>App: Payment Success UI (Unverified)

    Note over Backend, Telebirr: Phase 3: Settlement (Server Side)
    Telebirr->>Backend: POST /notify_url (Async Callback)
    Backend->>Backend: Verify SHA256WithRSA Signature
    Backend->>Backend: Update Order Status -> PAID
    Backend->>Backend: Calculate Split (95 ETB Vendor / 5 ETB Platform)
    Backend->>Telebirr: POST /merchant/transfer (Disbursement)
```

3.3 Data Element Dictionary

The following data elements are transmitted during the transaction lifecycle:

field_name	Type	Source	Destination	Description
`appId`	String	Backend	Telebirr	Unique Application ID assigned by Ethio Telecom.
`outTradeNo`	String	Backend	Telebirr	Unique Transaction Reference generated by Suuq.
`totalAmount`	String	Backend	Telebirr	Transaction Value (in ETB). Anti-tamper source.
`sign`	String	Backend	Telebirr	`SHA256WithRSA` signature of the payload.
`payload`	JSON	Backend	App	The signed object containing `prepayId` for the SDK.
`notify_url`	URL	Backend	Telebirr	The endpoint (`/api/payments/telebirr-callback`) receiving the final status.

4. Business Logic & Merchant of Record

4.1 Merchant of Record (MoR) Model

Suuq S acts as the single point of entry for funds to ensure compliance and trust.

- **Inflow:** 100% of the cart value is debited from the customer to the "Suuq S" Master Merchant Account.
- **Ledgering:** An internal immutable ledger (`telebirr_transaction` table) records the Fabric Reference ID against the local Order ID.
- **Outflow (Disbursement):** Upon successful callback verification:
  1. The system calculates the **Net Amount** (95% of the total order value) for the Vendor.
  2. The system calculates the **Commission Amount** (5% of the total order value) for Suuq S.
  3. A B2B transfer is executed immediately to the Vendor's registered Merchant ID for their 95% share.
  4. The remaining 5% is retained in the Master Merchant Account as platform revenue.

4.2 Data Integrity

- **Anti-Tampering:** Order amounts are stored server-side. The client cannot modify the amount sent to Telebirr; the signature is generated on the backend using the stored database value.
- **Double-Spending Prevention:** The system enforces standard Idempotency keys based on `OrderId`. Duplicate callbacks for the same ID are logged but not re-processed.

5. Threat Model & Security Mitigations

5.1 Identified Threats

Threat ID	Description	Severity	Mitigation Strategy
TR-01	**Replay Attacks**	High	Implementation of **Timestamp Nonce**. Requests outside a 60-second window (NTP Synced) are rejected by the Fabric Gateway.
TR-02	**Man-in-the-Middle (MITM)**	Critical	1. Strict SSL/TLS enforcement. 2. **RSA-2048 Encryption** (PKCS1 Padding) for sensitive payload data.
TR-03	**Unauthorized Disbursement**	Critical	Disbursement API endpoints are firewalled and accessible only via internal logic triggers, not exposed to the public API client.
TR-04	**SDK Reverse Engineering**	Medium	**ProGuard Obfuscation** applied to `com.huawei.ethiopia.pay.sdk.api.core.**` to hide internal logic in the Android APK.
TR-05	**Fake Callbacks**	High	All incoming callbacks to `notify_url` must pass **SHA256WithRSA** signature verification using the Telebirr Public Key.

5.2 Cryptographic Standards

- **Asymmetric Key:** RSA 2048-bit keys generated via OpenSSL.
- **Signature Algorithm:** `SHA256WithRSA`.
- **Payload Encryption:** AES / RSA hybrid approach where applicable (following Fabric V3 specs).

---

6. Functional & Non-Functional Requirements

6.1 Functional Requirements (FR)

User & Transaction Management:

1. **FR-01 (Transaction Initialization):** The system MUST generate a unique `outTradeNo` (Order ID) for every distinct payment attempt to prevent transaction collisions.
2. **FR-02 (Signature Generation):** The backend MUST sign all outgoing requests to the Telebirr Fabric gateway using the securely stored RSA-2048 Private Key.
3. **FR-03 (Callback Verification):** The system MUST intercept all `notify_url` POST requests and verify the `sign` parameter using the official Telebirr Public Key before processing.

Reconciliation & Integrity: 4. **FR-04 (Status Synchronization):** Upon successful signature verification, the system MUST atomically update the local Order status from `PENDING` to `PAID`. 5. **FR-05 (Manual Query):** The system MUST provide an Admin Interface to manually trigger the `queryOrder` Fabric API to resolve status discrepancies for "Stuck" transactions (e.g., network

timeout during callback). 6. **FR-06 (Double-Spend Protection):** The system MUST reject any callback with a `outTradeNo` that has already been marked as `PAID` in the ledger.

Disbursement Logic (Business Rules): 7. **FR-07 (Commission Calculation):** The system MUST automatically calculate the platform fee (5%) and the vendor payout (95%) immediately upon receipt of funds. 8. **FR-08 (Automated Payout):** The system MUST trigger a B2B transfer to the Vendor's registered merchant account for their 95% share only AFTER the initial customer debit is confirmed.

6.2 Non-Functional Requirements (NFR)

Security & Compliance:

1. **NFR-01 (Key Management):** Private Keys and App Secrets MUST be stored in environment variables (server-side) and NEVER exposed to the client-side code or Git repository.
2. **NFR-02 (Payload Encryption):** All sensitive PII (Personal Identifiable Information) in transmission MUST be encrypted using TLS 1.2+ in transit and AES-256 for database storage at rest.
3. **NFR-03 (Signature Standards):** Digital signatures must adhere strictly to `SHA256WithRSA` standards as mandated by Ethio Telecom.

Performance & Reliability: 4. **NFR-04 (Latency):** The payment initialization step (signing & token generation) MUST complete within 200ms to ensure a smooth user experience. 5. **NFR-05 (Availability):** The Payment Gateway Service endpoint MUST maintain 99.9% uptime during business hours. 6. **NFR-06 (Auditability):** All financial API interactions (Requests/Responses) MUST be logged to a centralized logging service (e.g., CloudWatch/PM2 Logs) with timestamps, IP addresses, and Request IDs for a minimum retention period of 1 year. 7. **NFR-07 (Obfuscation):** The Android Mobile Application MUST utilize ProGuard rule sets to obfuscate the payment SDK package `com.huawei.ethiopia.pay.sdk.**` to prevent reverse engineering.

---

7. Configuration Details (For Auditor Reference)

- **Environment:** Production
- **Backend Framework:** NestJS (Node.js)
- **Callback IP Allowlist:** Configured to accept specific Telebirr Gateway ranges (if applicable) or validated strictly via Signature.
- **NTP Server:** `pool.ntp.org` (Ensures synchronized timestamps for API requests).