System Description: Remote ID Authentication Server

Code	Layer Name	Purpose
[L1]	**External Actors**	Represents trusted external clients: the Authorized Registrar and Observer App.
[L2]	**API and Routing Layer**	Exposes HTTPS endpoints and dispatches requests to functional handlers.
[L3]	**Core Application Logic Layer**	Contains domain logic for drone provisioning and message verification.
[L4]	**Data Access and Storage Layer**	Manages persistent drone credentials and cache-based key retrieval.
[L5]	**Cryptographic Services Layer**	Provides cryptographic primitives and token/certificate verification logic.

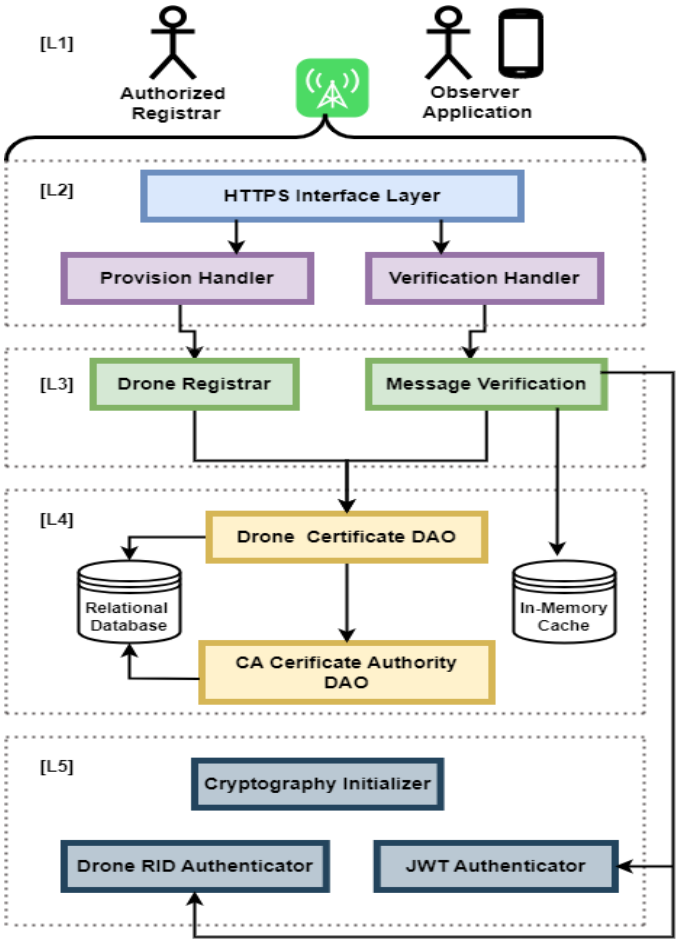

Fig. 1: Authentication Server structure

The Secure Remote ID Authentication Server is a multi-layer backend system designed for cryptographically secure drone provisioning and real-time identity verification. Its architecture is structured across five discrete layers, labeled [L1] through [L5], each serving a distinct role in the system's trust and control model.

In the External Interface Layer [L1], two actors interface with the server: the Authorized Registrar, which submits provisioning requests for drones, and the Observer Application, which relays signed messages broadcast by drones. These external clients interact with the system through the HTTPS Interface Layer in [L2].

The API and Routing Layer [L2] contains the Provision Handler and the Verification Handler, each responsible for processing a distinct class of incoming requests. The Provision Handler validates incoming drone registration requests and forwards valid data to the Drone Registrar in

[L3]. Malformed or duplicate entries are rejected. Upon successful registration, the handler returns a structured response with metadata confirming identity issuance.

The Verification Handler processes observer-submitted payloads and performs structural validation and token extraction. If a bearer token is not present or is improperly formatted, the request is rejected. Valid tokens are passed to Message Verification in [L3], which invokes the JWT Authenticator in [L5]. This component parses and verifies the token's signature using a configured RSA public key and ensures its validity against embedded claims.

To verify drone identity, Message Verification parses the incoming message to extract fields such as Drone UUID, Authentication Data, and Drone Signature. If any required fields are missing, the request is terminated. Otherwise, the drone's public key is retrieved from the In-Memory Cache in [L4]. On cache miss, the Drone Certificate DAO is queried to load the key from the Relational Database. The Drone RID Authenticator in [L5] then validates the digital signature over the authentication data using the Ed25519 algorithm. This signature check ensures that the message originated from a drone in possession of the private key corresponding to its registered certificate.

After completing both observer and drone validations, Message Verification maps the outcome to an internal result code representing one of several verification states: valid, invalid, expired, unknown key, revoked, or unknown identifier. The result is returned to the Verification Handler, which formats a structured response containing observer identity, drone UUID, result code, and interpretation text.

The Core Application Layer [L3] includes both the Drone Registrar, which handles keypair generation and certificate issuance, and Message Verification, which performs end-to-end authentication logic. The Drone Registrar consults the Drone Certificate DAO in [L4] to persist the credential set, and obtains the signing key from the CA Certificate Authority DAO to finalize certificate issuance.

The Data Access and Storage Layer [L4] supports both provisioning and verification paths. The Drone Certificate DAO manages persistent storage of drone certificates and keys and interfaces with the Relational Database. The CA Certificate Authority DAO provides access to the CA’s signing material. The In-Memory Cache optimizes public key retrieval for repeated verification flows, reducing lookup latency during authentication.

The Cryptographic Services Layer [L5] houses the components that enforce digital trust. The Cryptography Initializer configures the cryptographic environment, including provider registration and key parsing. The Drone RID Authenticator performs Ed25519 signature validation on authentication data, while the JWT Authenticator verifies RS256-signed identity tokens. These components are stateless and invoked on demand by components in [L3].

Requests pass from interface components in [L2], through business logic in [L3], to supporting infrastructure in [L4] and [L5]. Each verification result is based on deterministic logic involving signature validation, key resolution, and trust chain anchoring. This structure ensures verifiability,

isolation of responsibility, and cryptographic assurance across all provisioning and identity verification operations.

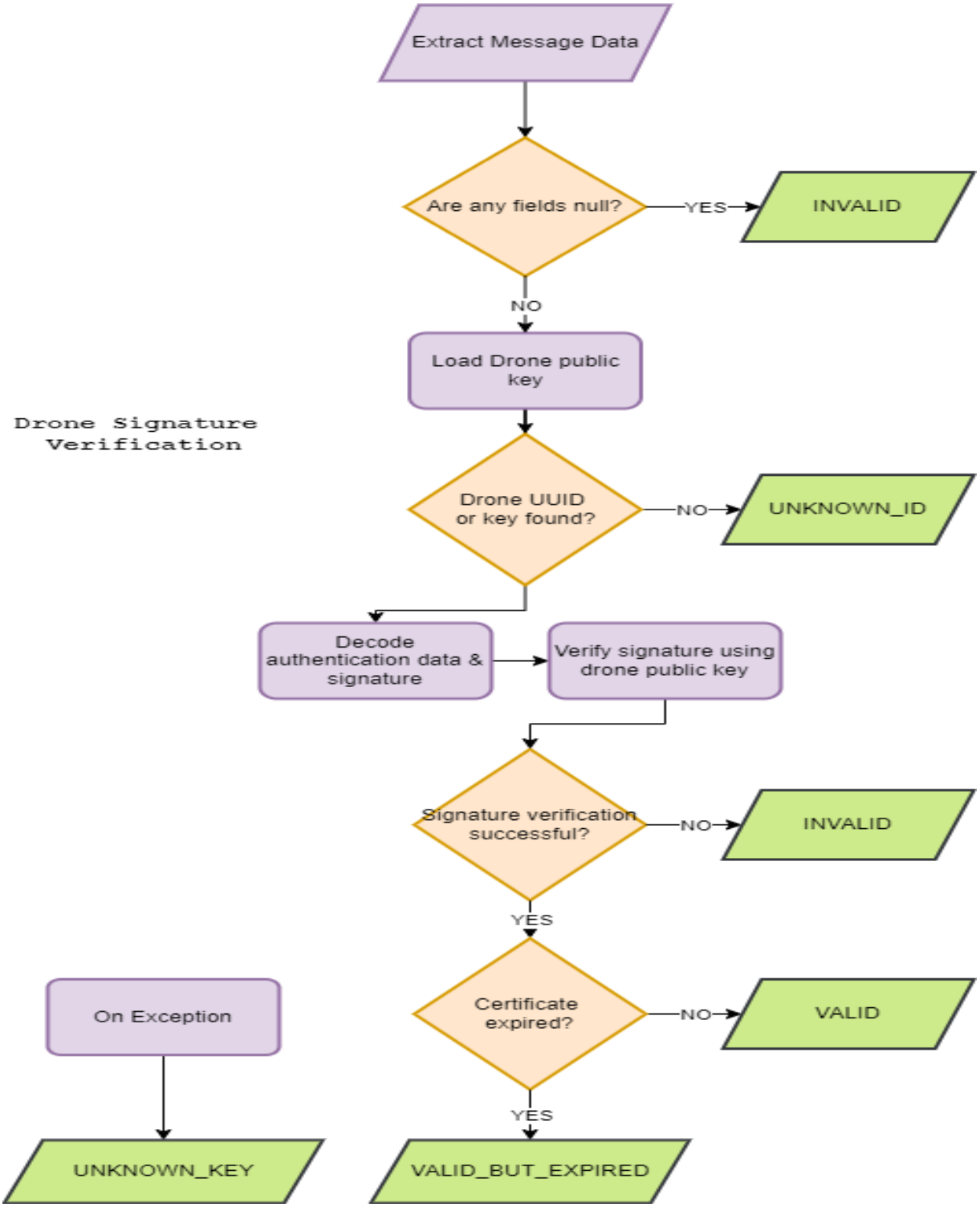

Fig. 2: Flow Chart of Drone Signature Verification.

Drone signature verification steps:

The drone signature verification flowchart (Fig. 2) details the sequential logic executed by the server to authenticate the integrity and origin of a Remote ID message broadcast by a drone. The process begins with extracting the message data received from the observer. The first validation step checks whether any required fields in the message are null or missing. If any field is null, the message is immediately marked as INVALID. If all required fields are present, the server proceeds to load the drone's public key, which is associated with a unique identifier (UUID).

The next step verifies whether the drone UUID or the corresponding public key is found in the server's records. If not, the verification process terminates with an UNKNOWN_ID result. If the key is found, the server proceeds to decode the authentication data and the associated signature from the message. It then verifies the signature using the drone's public key. If the signature verification fails, the message is again marked as INVALID.

If the signature is successfully verified, the server checks whether the drone's certificate is still valid. If the certificate has expired, the server responds with VALID_BUT_EXPIRED, acknowledging a valid signature but flagging outdated credentials. If the certificate is still valid, the message is marked as VALID. Any exceptions encountered during these operations—such as unexpected errors or cryptographic failures—lead to an UNKNOWN_KEY status. This multi-stage process ensures that only authenticated and untampered messages from recognized drones are accepted, supporting secure and trustworthy Remote ID operations.

Observer Verification Flowcahrt:

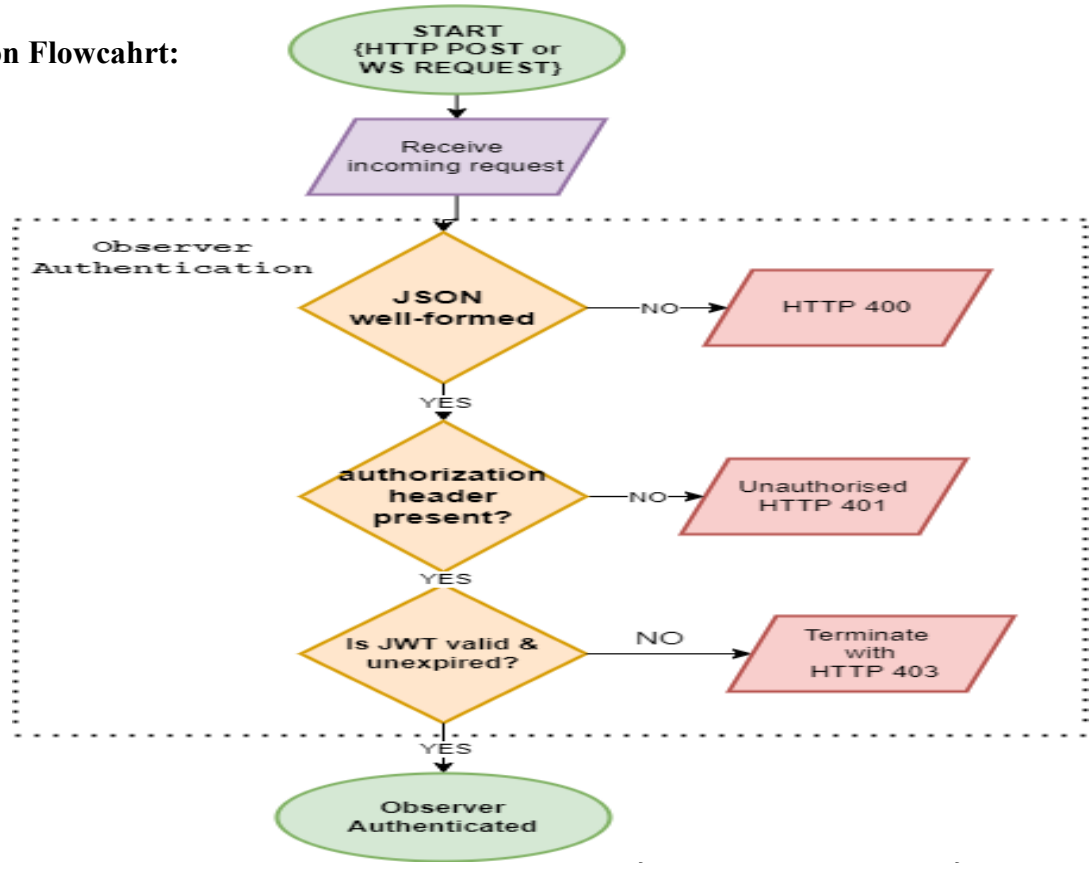

Fig. 3: Flow Chart Observer Verification.

The observer authentication flowchart (Fig. 3) outlines the step-by-step process used by the Remote ID authentication server to validate incoming HTTP POST or WebSocket (WS) requests from observers. The process begins with the receipt of an incoming request, which is then subjected to a series of validation checks. The first validation step ensures that the received payload is a well-formed JSON object. If this check fails, the server immediately responds with an HTTP 400 Bad Request error, indicating a malformed input. If the JSON is valid, the process continues by checking for the presence of an authorization header in the request. Absence of this header triggers an HTTP 401 Unauthorized response, indicating that authentication credentials are missing.

Once the authorization header is confirmed, the next step involves validating the JWT (JSON Web Token) included in the header. The server checks whether the token is valid and unexpired. If the JWT is invalid or expired, the server terminates the authentication process and returns an HTTP 403 Forbidden response. Only when all these checks are passed successfully is the observer considered authenticated, allowing access to further resources or operations. This multi-step validation process ensures secure and standards-compliant observer verification within the Remote ID authentication framework.