API Specification

1. **Cryptographic API Development:**
   - Design and develop an API with the following specific functionalities:
     Key Generation Endpoint:
     - **Description:** This endpoint allows users to generate cryptographic keys.
     - **Request:** POST `/generate-key`
       - Body: `{ "key_type": "AES", "key_size": 256 }`
       - Parameters:
         - `key_type`: Specifies the type of encryption (e.g., AES, RSA).
         - `key_size`: Specifies the size of the key in bits.
     - **Response:**
       - `{ "key_id": "12345", "key_value": "base64-encoded-key" }`
         - `key_id`: A unique identifier for the generated key.
         - `key_value`: The cryptographic key encoded in Base64 for portability.
   - **Encryption Endpoint:**
     - **Description:** This endpoint encrypts plaintext messages using a specified key and encryption algorithm.
     - **Request:** POST `/encrypt`
       - Body: `{ "key_id": "12345", "plaintext": "message-to-encrypt", "algorithm": "AES" }`
       - Parameters:
         - `key_id`: Identifies the key to be used for encryption.
         - `plaintext`: The message that needs encryption.
         - `algorithm`: Specifies the encryption algorithm to use (e.g., AES, RSA).
     - **Response:**
       - `{ "ciphertext": "base64-encoded-ciphertext" }`
         - `ciphertext`: The encrypted message in Base64 format.
   - **Decryption Endpoint:**
     - **Description:** This endpoint decrypts encrypted messages back into plaintext.
     - **Request:** POST `/decrypt`
       - Body: `{ "key_id": "12345", "ciphertext": "base64-encoded-ciphertext", "algorithm": "AES" }`
       - Parameters:
         - `key_id`: Identifies the key to be used for decryption.

  - `ciphertext`: The message that needs decryption.
  - `algorithm`: Specifies the encryption algorithm to use (e.g., AES, RSA).
- **Response:**
  - `{ "plaintext": "original-message" }`
    - `plaintext`: The decrypted original message.

2. **Hashing API Development:**
   - Design and develop an API for hashing and verifying data integrity with the following endpoints:
     Hash Generation Endpoint:
     - **Description:** This endpoint generates a hash for the given data using a specified hashing algorithm.
     - **Request:** POST `/generate-hash`
       - Body: `{ "data": "message-to-hash", "algorithm": "SHA-256" }`
       - Parameters:
         - `data`: The input data to be hashed.
         - `algorithm`: The hashing algorithm to use (e.g., SHA-256, SHA-512).
       - **Response:**
         - `{ "hash_value": "base64-encoded-hash", "algorithm": "SHA-256" }`
           - `hash_value`: The generated hash in Base64 format.
           - `algorithm`: The hashing algorithm used.
   - **Hash Verification Endpoint:**
     - **Description:** This endpoint verifies if the given hash matches the data.
     - **Request:** POST `/verify-hash`
       - Body: `{ "data": "message-to-verify", "hash_value": "base64-encoded-hash", "algorithm": "SHA-256" }`
       - Parameters:
         - `data`: The original input data.
         - `hash_value`: The hash to be verified.
         - `algorithm`: The hashing algorithm to use.
       - **Response:**
         - `{ "is_valid": true, "message": "Hash matches the data." }`
           - `is_valid`: A boolean indicating whether the hash matches the data.
           - `message`: Additional information about the result.