Here’s the rewritten **Orgo v2 Blueprint Section 5 - Security Configuration** with the requested corrections integrated:

### Section 5: Security Configuration

This section defines the measures and configurations necessary to ensure the security, privacy, and compliance of Orgo. The updates integrate pseudonymization mapping, compliance-friendly audit trails, and advanced encryption protocols.

### 5.1 Purpose of Security Configuration

**Objective**: Safeguard email communication and workflows against unauthorized access, data breaches, and privacy violations.

**Outcome**: A secure platform that meets organizational and regulatory requirements (e.g., GDPR, HIPAA).

### 5.2 Core Security Features

#### Email Encryption

* **TLS (Transport Layer Security)**: Encrypts email transmission between Orgo and the email server.
  + **Mandatory Scenarios**: All general email communications (e.g., maintenance requests).
  + Example Configuration:

smtp:

host: "smtp.organization.com"

port: 587

tls: true

username: "orgo@organization.com"

password: "securepassword"

* **PGP (Pretty Good Privacy)**: Encrypts sensitive email content using public-private key pairs.
  + **Mandatory Scenarios**: Highly sensitive workflows (e.g., harassment reports, patient data).
  + Example Workflow:
    - Sender encrypts the email with the recipient’s public key.
    - Only the recipient can decrypt it using their private key.

#### Role-Based Access Control (RBAC)

Ensures users can only access workflows and data relevant to their roles.

* **Example Roles**:
  + Administrator: Full access to workflows and logs.
  + HR Team: Access to harassment reports.
* **Implementation Example**:

CREATE TABLE roles (

role\_id SERIAL PRIMARY KEY,

role\_name VARCHAR(50)

);

CREATE TABLE user\_roles (

user\_id INT,

role\_id INT,

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (role\_id) REFERENCES roles(role\_id)

);

#### Sensitive Data Anonymization

Removes identifying metadata (e.g., sender email, name) for sensitive workflows.

* **Pseudonymization Mapping**:

pseudonyms:

- id: user-1234

real\_id: employee@organization.com

* + Mapping files should be securely stored with access limited to authorized personnel only.
* **Audit Example**:

[2024-11-24T10:00:00] Workflow: Harassment Report | Reporter: user-1234

#### Secure Data Storage

* **Encryption Standards**: AES-256 for emails, logs, and database content.
* **Database Example**:

CREATE EXTENSION pgcrypto;

INSERT INTO sensitive\_data (encrypted\_column)

VALUES (pgp\_sym\_encrypt('Sensitive Content', 'encryption\_key'));

#### Multi-Factor Authentication (MFA)

Adds a second authentication layer for administrators accessing Orgo’s backend.

* **Tools**: Google Authenticator, Authy (TOTP).

#### Audit Logging and Monitoring

Logs all system actions for transparency and compliance.

* **Examples of Logged Events**:
  + Email parsing and routing decisions.
  + Access attempts (successful or failed).
* **Log Example**:

{

"timestamp": "2024-11-24T12:34:56",

"action": "Email Routed",

"email\_id": "12345",

"from": "secretary@organization.com",

"to": "maintenance@organization.com",

"status": "success"

}

### 5.3 Compliance with Privacy and Security Standards

#### GDPR (General Data Protection Regulation)

* Pseudonymization and consent tracking for sensitive workflows.
* Example: Anonymized harassment reports ensure no identifying details are exposed to unauthorized personnel.

#### HIPAA (Health Insurance Portability and Accountability Act)

* Secure handling of patient information in healthcare workflows.
* Example: Encrypted email transmission and storage for patient data.

#### Other Regional Regulations

Customizable for country-specific laws (e.g., CCPA, PIPEDA).

### 5.4 Security Workflow Examples

#### Harassment Reporting Workflow

1. Employee sends a sensitive email to report@organization.com.
2. Orgo anonymizes the sender (e.g., user-1234) and encrypts the email content using PGP.
3. The email is routed to HR and legal advisors, with access restricted to designated roles.
4. All actions are logged for compliance audits.

#### Maintenance Request Workflow

1. Secretary sends a request to emergency@organization.com.
2. Orgo logs the request, routes it to the maintenance team, and encrypts task details.
3. Only team members assigned to the task can access the details.

### 5.5 Security Testing and Validation

* **Penetration Testing**: Periodic tests to identify vulnerabilities in email handling, routing, and storage.
* **Data Integrity Checks**: Validate that encrypted data can be decrypted only by authorized users.
* **Access Control Validation**: Simulate role-based access to verify unauthorized users cannot access sensitive workflows.

### 5.6 Encryption Protocols

Example for AES-256 key storage and usage:

from Crypto.Cipher import AES

key = b'Sixteen byte key'

cipher = AES.new(key, AES.MODE\_CFB)

encrypted = cipher.encrypt(b'Sensitive Data')

### 5.7 Deliverables

* **Security Policies**: Documentation for RBAC, encryption standards, and log retention.
* **Configuration Files**: Predefined configurations for TLS, PGP, and database encryption.
* **Test Reports**: Logs from penetration testing and role-based access simulations.

### Summary

This section ensures Orgo’s workflows and data are protected through robust security measures, including encryption, pseudonymization, and access control. By complying with global privacy standards and implementing rigorous logging, Orgo provides a secure and reliable platform for handling sensitive communication.