Project Proposal Format

Project Title: Secure Authentication System

Team Members:

Aidos (GitHub: @Aidos228) - Role: Backend/Crypto

Arsen (GitHub: @TheSuxarick) - Role: Backend/Crypto

Adil (GitHub: @KIAMOOO) - Role: Frontend/Integration

Project Option: Option 3

Brief Description:

This project is a secure authentication web application built with Flask that demonstrates robust security practices. It features a comprehensive user authentication flow including registration, login with Two-Factor Authentication (TOTP), and password reset functionality via email using cryptographic tokens. The system is designed to showcase the practical implementation of various cryptographic algorithms for data protection, session management, and secure communications.

Cryptographic Components:

- Symmetric Encryption: AES-GCM (256-bit) is used for securing sensitive data at rest and in transit where applicable, ensuring confidentiality and integrity.
- Asymmetric Encryption: RSA-OAEP (2048-bit) is employed for secure key exchange and encrypting small amounts of highly sensitive data.
- Digital Signatures: RSA-PSS is used for generating and verifying digital signatures to ensure non-repudiation and data authenticity, particularly in token validation.
- Key Exchange: Diffie-Hellman key exchange protocol is simulated/implemented to demonstrate secure shared secret establishment over an insecure channel.
- Hashing: SHA-256 is utilized for integrity checks and generating digests for signatures. Bcrypt is used for secure password hashing with salt to protect user credentials.
- Token Management: JSON Web Tokens (JWT) are used for stateless authentication and secure session management, signed using HMAC or RSA.
- TOTP: Time-based One-Time Passwords (RFC 6238) are implemented for 2FA using HMAC-SHA1.

Architecture Overview:

The system follows a classic client-server architecture:

- Client Layer: A web browser interface built with HTML/CSS (rendered via Flask templates) that interacts with the user. It also integrates with external TOTP authenticator apps (like Google Authenticator) for 2FA.
- Application Server: A Flask-based backend that handles HTTP requests, business logic, and routing. It orchestrates the authentication flow, enforces security policies, and manages user sessions using both Flask sessions and JWTs.
- Cryptographic Layer: A dedicated module within the application that encapsulates all cryptographic operations (encryption, decryption, signing, hashing) to ensure a clean separation of concerns and reusable security logic.
- Data Persistence: A lightweight JSON-based storage (users.json) is used for user account management in this educational context.