🔐 Cipher Shield

Project Objective: To build a robust, real-world secure chat system leveraging hybrid cryptography, multi-layered authentication, and threat detection through machine learning.

🔹 1. CIA Triad – Core Security Principles

🛡️ Confidentiality

We ensure that only the intended sender and receiver can read the message using:

- **AES (CBC mode)** for encrypting message contents securely.
- **ECC (Elliptic Curve Cryptography)** for lightweight but secure encryption of session keys.
- **TLS (Transport Layer Security)** for protecting data in transit between client and server.
- **Encrypted file storage** for all user-uploaded files (e.g., images, documents).

🔐 Integrity

We preserve the trustworthiness of data with:

- **Digital Signatures** using RSA/ECC that verify the message hasn't been altered.
- **SHA-256** hashing of both passwords and message digests before database storage or transmission.
- Automatic **signature verification on message receipt** ensures no tampering during transit.

⚙️ Availability

We guarantee system responsiveness and uptime through:

- Scalable **Flask/FastAPI APIs** optimized for minimal latency and timeout recovery.
- Anomaly-based traffic monitoring to prevent DoS-style overloads.
- Message queue retries for failed transmissions and backup recovery options.

🔹 2. Threat Models & Mitigation Strategy

Threat	**Description**	**Defense Mechanism**
MITM (Man-in-the-Middle)	Attacker intercepts and possibly alters messages between users.	End-to-end encryption + HTTPS + ECC-based key exchange.
Replay Attacks	Captured packets are resent to mimic valid communication.	Nonce values and timestamps in every request; server discards expired or reused tokens.

Threat	Description	Defense Mechanism
Brute Force Attacks	Attacker attempts millions of password or key combinations.	Passwords hashed with SHA-256 + salted; login limited by rate-limiting & CAPTCHA.
SQL Injection/XSS	Exploiting unfiltered input to access database or run malicious scripts.	Input sanitization, ORM usage, and secure headers like CSP, X-Content-Type-Options.
Insider Threats	A legitimate user (admin/dev) misuses system access.	Audit logging for every admin action, RBAC (role-based access control).
Botnets / Message Flooding	Automated scripts send bulk traffic to crash or confuse the system.	ML-based anomaly detection using SVM or RandomForest to analyze frequency/IP deviation.
Ransomware Attacks	Malicious users upload infected files or exploit vulnerabilities to encrypt platform data and demand ransom.	File type validation, MIME checking, sandboxed file handling, encrypted storage, and automated encrypted backups.

🧨 Ransomware Countermeasures (Expanded)

- **File Isolation:** Every uploaded file is scanned for binary signatures and stored outside public directories.
- **Behavioral Monitoring:** If a user uploads multiple large or unreadable files rapidly, the system logs it and flags it for review.
- **Auto-Backup:** System performs daily encrypted backups of user data & logs. If a ransomware lockout happens, data recovery is instant.
- **File Decryption Guard:** If files are detected with strange extensions (e.g., .locky, .crypt), they are flagged and stored in read-only mode.

🔹 3. Security Scope (What We Secure)

✅ Components Under Security Control:

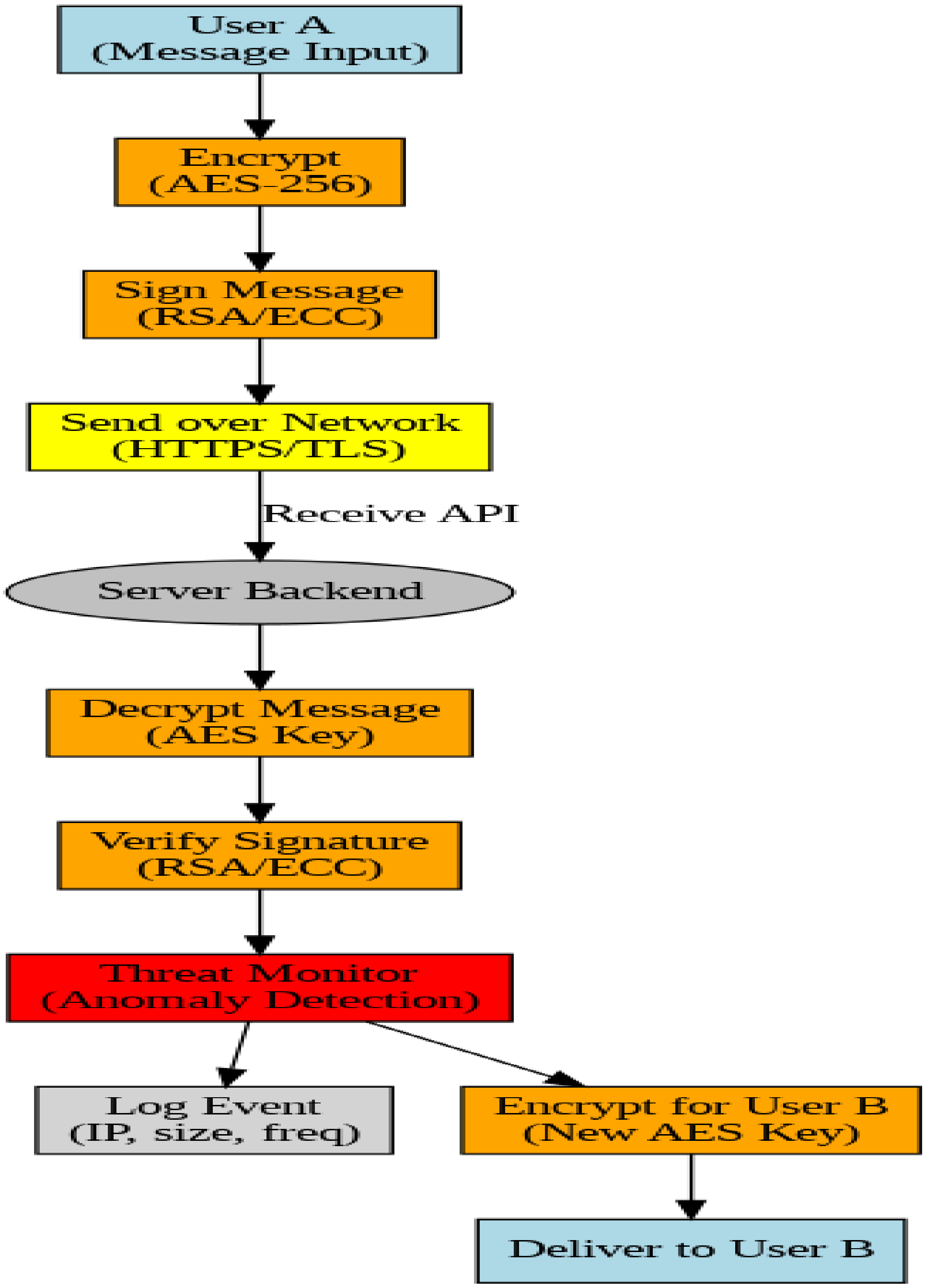
- **User credentials:** Stored as SHA-256 hashes + salting. No plain-text storage.
- **Text & multimedia messages:** Encrypted with AES-256. Transport secured with HTTPS.
- **File uploads:** MIME-checked, scanned, and stored encrypted.
- **Session data:** Token-based session storage with expiration + refresh cycle.
- **Communication logs:** Encrypted logs for login time, IPs, message counts for anomaly detection.

❌ **Components Out of Scope (for now):**

- Full mobile integration with secure key sync
- Hardware-backed key modules (HSMs, TPM)
- Encrypted peer-to-peer voice/video calling
- Fully anonymized routing (e.g., Tor-based)

🔹 4. Communication Flow & Security Layers

🔄 **User-to-User Message Lifecycle:**


User A (Message Input)
Encrypt (AES-256)
Sign Message (RSA/ECC)
Send over Network (HTTPS/TLS)
Receive API
Server Backend
Decrypt Message (AES Key)
Verify Signature (RSA/ECC)
Threat Monitor (Anomaly Detection)
Log Event (IP, size, freq)
Encrypt for User B (New AES Key)
Deliver to User B

At every step:

- **Logs** are generated (time, size, source IP, pattern)
- **Signature verification** ensures authenticity
- **Threat detection** module constantly checks for anomalous spikes or suspicious traffic

---

🧠 Optional Additions for Later:

- OTP-based MFA layer + biometrics placeholder
- Blockchain-based message integrity proof-of-record
- UI dashboard for live cryptanalysis + threat alerts
- Admin control to simulate brute-force/DoS and see system response

---

✅ Summary

This system is designed to **replicate real-world secure communication** infrastructure. It integrates:

- Practical hybrid cryptography
- Secure data management
- AI-driven threat detection
- Simulated vulnerability assessment

This makes it not just a secure chat — but a learning playground for cryptanalysis, attack simulation, and advanced protocol design.
