PROBLEM STATEMENT:

Protecting User Password Keys at Rest

With increasing data breaches and cyber-attacks, securing user password keys stored at rest has become critical. Traditional methods of storing passwords, even when hashed, can be vulnerable if the underlying storage is compromised. An attacker gaining access to the disk can potentially retrieve and misuse these keys. Therefore, a robust system is required to ensure that password keys remain secure even if the storage medium is compromised.

Unique Idea Brief:

The unique idea behind this project is to implement a multi-layered security approach to protect user password keys at rest. This includes encrypting the keys with a strong symmetric encryption algorithm (AES), securely managing the encryption keys using asymmetric encryption (RSA), enforcing strict access controls with multi-factor authentication (MFA), and maintaining comprehensive logging for audit purposes. By integrating these components, the system provides robust security for password keys stored on the disk.

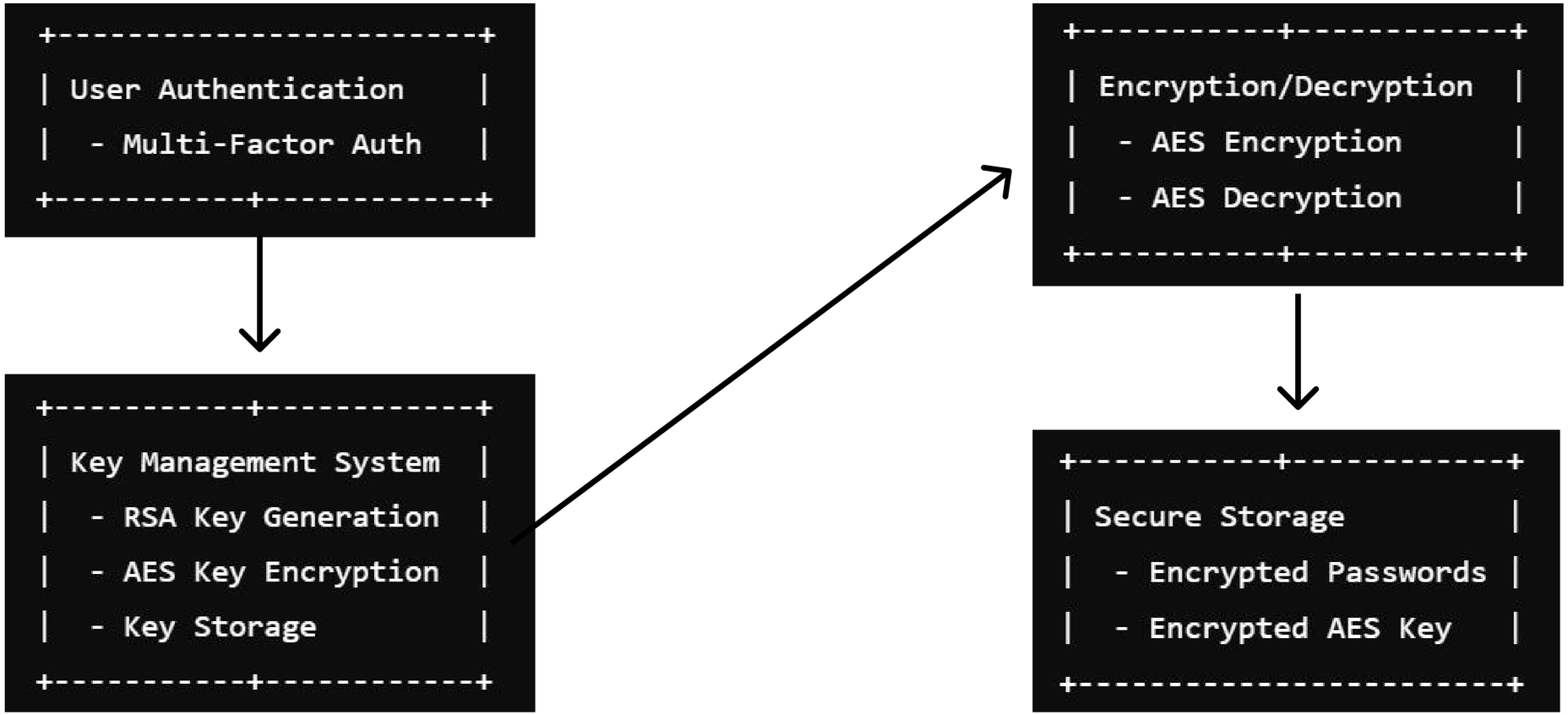
Features Offered:

- **Strong Encryption:** Uses AES for encrypting password keys to ensure confidentiality.
- **Secure Key Management:** Handles key generation, storage, and rotation using RSA encryption.
- **Access Control:** Enforces strict access policies with multi-factor authentication.
- **Auditing and Logging:** Maintains logs for all key management operations and alerts for suspicious activities.
- **Scalability:** The system is designed to be scalable and can be integrated into larger security infrastructures.

Process Flow:

1. **Key Generation:** Generate a strong AES key using PBKDF2HMAC and store it securely.
2. **Password Encryption:** Encrypt user password keys using the AES key and store the encrypted passwords.
3. **Password Decryption:** Decrypt the stored encrypted passwords when needed using the AES key.
4. **Key Management:** Generate RSA key pairs, encrypt the AES key with the RSA public key, and securely store the encrypted AES key.
5. **Access Control:** Implement multi-factor authentication to control access to the key management system.
6. **Auditing and Logging:** Log all key management operations and monitor for suspicious activities.

Architecture Diagram:



Technologies Used:

Python: Programming language for implementing the system.

Cryptography: Python library for cryptographic operations (e.g., cryptography, pycryptodome).

pyotp: Library for implementing multi-factor authentication.

Logging: Standard Python logging library for auditing.

Team Members and Contribution:

Sarvagya: Sole contributor responsible for the entire project, including conceptualization, design, implementation, testing, and documentation along with the help of online available resources and libraries.

Conclusion:

This project provides a comprehensive solution for protecting user password keys at rest by integrating strong encryption, secure key management, multi-factor authentication, and auditing. The multi-layered security approach ensures that even if the disk is compromised, the password keys remain secure. The system is designed to be scalable and adaptable, making it a valuable addition to any organization's security infrastructure.
4o