Project Title: CRYPTOVAULT: A GUI-BASED TOOLKIT FOR HYBRID ENCRYPTION AND CUSTOM ALGORITHM DESIGN

Student ID	Student Name
1276327	ASMA UL HUSNA
1276026	AYE KYI KYI CHO

PROJECT PROPOSAL

1. Problem/Project Definition

Despite the growing need for data security, existing encryption tools often fail to balance usability, flexibility, and education. Many solutions are either too complex for non-experts or too rigid to allow experimentation with custom cryptographic workflows. Additionally, while classical cyphers (e.g., Caesar, XOR) are essential for learning, they are rarely integrated with modern standards (e.g., AES, RSA) in a single platform. To bridge these gaps, this project develops CryptoVault, a user-friendly GUI toolkit that simplifies encryption/decryption while enabling custom algorithm design. By combining intuitive controls with support for hybrid and historical cyphers, CryptoVault serves both as a practical security tool and an educational resource for understanding cryptography.

2. Significance of the Project

This project holds substantial value for both security professionals and students by bridging theoretical cryptography with practical implementation. Unlike command-line tools that dominate the space, the GUI interface lowers the barrier to entry while maintaining robust security through proper implementations of AES and RSA. The custom algorithm builder represents a unique contribution, enabling users to experiment with multi-step cryptographic transformations - a feature absents in most existing tools. For organizations, it provides an auditable encryption workflow with complete operation history.

3. Brief Literature Review

Modern cryptography builds upon established standards like NIST's FIPS 197 (AES) and PKCS#1 (RSA), which form the foundation of this project's secure implementations. Academic research in cryptographic education (e.g., Paar & Pelzl's "Understanding Cryptography") inspired the inclusion of didactic ciphers alongside production algorithms. The project synthesizes concepts from OpenSSL's key management, GPG's file encryption approach, and cryptanalysis principles into a cohesive visual interface, while addressing documented usability challenges in existing tools.

4. Comparison with Previous Work

Where traditional tools like OpenSSL focus on single-algorithm operations through command-line interfaces, this project introduces three key advancements: (1) visual algorithm composition through its custom builder, (2) integrated cryptographic utilities (password generation, hashing, analysis), and (3) session persistence. Unlike web-based encryption tools, it maintains complete offline operation, eliminating cloud dependency risks. The dual implementation of both educational and real-world algorithms provides a graduated learning path absent in professional tools like VeraCrypt or academic tools like Cryptool.

5. Technical Architecture

The application employs a modular architecture with four core components: the cryptographic engine (leveraging Python's cryptography library), UI controller (Tkinter), operation history tracker, and preset manager. Symmetric operations use AES-CFB mode with automatic key length validation, while asymmetric operations implement RSA with OAEP padding (SHA-256). The custom algorithm builder treats each transformation (e.g., Caesar shift, XOR) as a pluggable component, executed sequentially during encryption and reversed for decryption. This design enables future expansion without architectural changes.

6. Usage Documentation

Users navigate through four primary tabs: Symmetric (AES/keyed ciphers), Asymmetric (RSA), Custom Builder, and Utilities. The workflow typically begins with key generation (automated for RSA), followed by text input and algorithm selection. The custom builder allows step addition/removal via intuitive controls, with parameters validated in real-time. All operations automatically log to the history panel, which supports CSV export. Notable features include password strength visualization, text entropy analysis, and one-click session persistence that saves all keys, texts, and algorithm configurations.

7. Security Considerations

While implementing industry-standard cryptographic primitives, the application enforces several security measures: AES keys are validated for correct length (128/192/256-bit), RSA keys use 2048-bit minimum length with proper padding, and all operations occur in memory without temporary file storage. However, users must protect private keys and session files, as the application doesn't implement secure storage mechanisms. The example ciphers (Caesar, XOR) are clearly marked as educational tools unsuitable for sensitive data, preventing accidental misuse.

8. Installation & Requirements

The suite requires Python 3.8+ with dependencies installable via pip (cryptography≥41.0.0). It maintains compatibility across Windows, macOS, and Linux without platform-specific modifications. The standalone executable option (via PyInstaller) eliminates Python environment requirements for end-users. Memory requirements are minimal (500MB RAM recommended for large text operations), with no persistent disk usage beyond user-saved sessions and preset files.

9. Future Enhancements

Planned developments include file encryption capabilities (chunked processing for large files), secure key storage using operating system credential managers, and additional algorithms (ChaCha20, ECC). The roadmap also proposes network features for secure messaging and collaborative algorithm design. Community contributions will be facilitated through modular architecture, with particular interest in expanding the custom algorithm step library and localization support.

10.References

[1]. R. Pasarelski, K. Angelov, K. Postagian and S. Sadinov, "Implementation and Analysis of a Customized Encryption Algorithm in 5G Networks for Educational Purposes," *2023 4th International Conference on Communications.*

[2]. Stephen Hart [a], Andrea Margheri [a], Federica Paci [b], Vladimiro Sassone [a] , Riskio: A Serious Game for Cyber Security Awareness and Education.

[3]. E. Thambiraja, G. Ramesh, Dr. R. Umarani, A Survey on Various Most Common Encryption Techniques.

[4]. Ako Muhamad Abdullah, Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data.

[5]. By Dr. Prerna Mahajan & Abhishek Sachdeva IITM, India, A Study of Encryption Algorithms AES, DES and RSA for Security.