INTEL-UNNATI AI&ML JOURNAL
Number 2 (2024), pp. 1–7

Team N-Coders 2024

CRYPTOGRAPHY

Cryptography Simulation with `mbedTLS/OpenSSL` Library Usage and User Interaction

Aaron Thomas Blessen, Sreejith A, and Midhun Sreenivas

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: This project develops an interactive cryptographic simulation using `OpenSSL` in C. It provides a hands-on platform for understanding encryption, decryption, key generation, and cryptographic algorithms. The solution includes a C-based engine, a user-friendly interface, and comprehensive documentation, enhancing users' practical knowledge of cryptographic processes and digital security.

Keywords: Cryptography, Simulation, `OpenSSL`, Encryption, Decryption, Key Generation, Cryptographic Algorithms, C Programming, Hands-on Learning, Digital Security, User Interface, Documentation

1 Introduction

Cryptography is essential for securing digital communication and data, but its complexity often makes it difficult to understand. This project aims to create an interactive simulation using `OpenSSL` in C, allowing users to explore encryption, decryption, key generation, and various cryptographic algorithms. The project will deliver a C-based cryptography engine, a user-friendly interface (command-line or graphical) for easy interaction, and comprehensive documentation detailing cryptographic algorithms, usage instructions, and technical aspects. By providing a practical, hands-on learning platform, this project seeks to demystify cryptographic processes and enhance users' understanding of digital security mechanisms.

2 Literature Survey and Related Works

The field of cryptography is fundamental to securing digital communications and data. Understanding cryptographic techniques such as encryption, decryption, and key generation is crucial for protecting information in various applications. The OpenSSL library is prominent tool for implementing these techniques. [2] introduces about the security using Key Generation and Encryption into the project.A detailed idea about the encryption on the project was from the reference [4].Conversely, OpenSSL is a comprehensive library utilized in high-performance applications, with [5] showcasing its effectiveness in optimizing secure communications. The cryptographic algorithm and different types of cryptography was from the surveys about [3] and [1]

Building on these insights, this project aims to develop an interactive cryptographic simulation using OpenSSL in C. The project integrates a robust cryptographic engine with an intuitive user interface and comprehensive documentation. This approach draws on the strengths of OpenSSL, as well as the educational benefits of interactive platforms like CrypTool. The resulting tool will enable users to explore encryption, decryption, key generation, and various cryptographic algorithms, significantly advancing digital security education by providing a practical, hands-on learning environment.

3 Contribution

The exploration of various cryptographic libraries and optimization techniques, along with the analysis of existing educational tools, has yielded valuable insights for our project. By leveraging the strengths of OpenSSL, we aim to create an interactive simulation that enhances users' understanding and practical knowledge of cryptographic techniques.

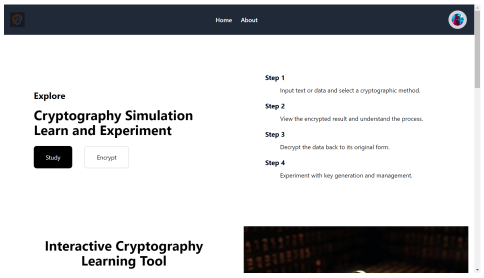
Our project will feature a robust cryptographic engine, an intuitive user interface, and comprehensive documentation, providing a platform for exploring encryption, decryption, key generation, and cryptographic algorithms. This contribution significantly advances digital security education by offering a practical, hands-on learning environment, thereby enhancing users' proficiency in cryptographic processes.

4 System Design

The system design of the cryptography simulation project is crucial for ensuring that the software is robust, secure, and user-friendly. This section outlines the overall architectural design, details the design of the cryptography engine, and explains the design of the user interface.

4.1 Architectural Design

The architectural design of the cryptography simulation project is built around a modular approach, where each component of the system is encapsulated into distinct modules to enhance maintainability and scalability. The core components include the cryptography engine, user interface, and a controller that mediates interactions between the engine and the interface. The cryptography engine handles all cryptographic operations, while the user interface facilitates user interaction. This separation of concerns ensures that the system can be easily extended or modified without impacting other components. Additionally, the use of the OpenSSL library ensures that the cryptographic operations adhere to industry standards for security and performance.



Explore

Cryptography Simulation
Learn and Experiment

Step 1
Input text or data and select a cryptographic method.

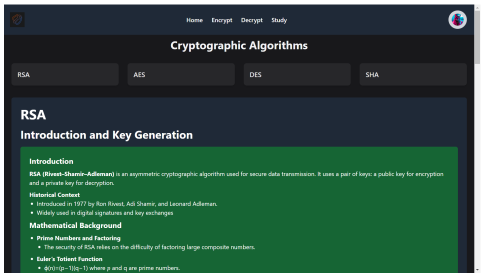
Step 2
View the encrypted result and understand the process.

Step 3
Decrypt the data back to its original form.

Step 4
Experiment with key generation and management.

Interactive Cryptography
Learning Tool

Figure 1: Home Page



Cryptographic Algorithms

RSA

Introduction and Key Generation

Introduction

RSA (Rivest–Shamir–Adleman) is an asymmetric cryptographic algorithm used for secure data transmission. It uses a pair of keys: a public key for encryption and a private key for decryption.

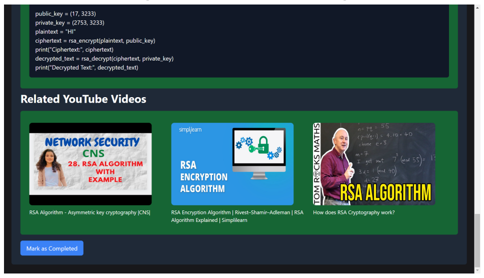
Historical Context

- Introduced in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman.
- Widely used in digital signatures and key exchanges

Mathematical Background

- **Prime Numbers and Factoring**
  - The security of RSA relies on the difficulty of factoring large composite numbers.
- **Euler's Totient Function**
  - ϕ(n)=(p−1)(q−1) where p and q are prime numbers.

Figure 2: Study Page 1



```
public_key = (17, 3233)
private_key = (2753, 3233)
plaintext = "HI"
ciphertext = rsa_encrypt(plaintext, public_key)
print("Ciphertext:", ciphertext)
decrypted_text = rsa_decrypt(ciphertext, private_key)
print("Decrypted Text:", decrypted_text)
```

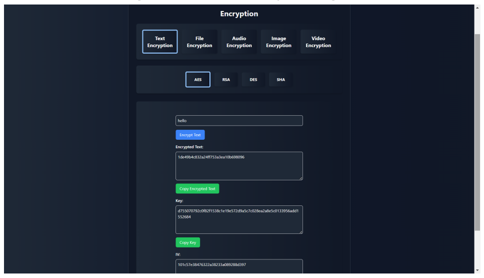
Related YouTube Videos

RSA Algorithm - Asymmetric key cryptography |CNS|

RSA Encryption Algorithm | Rivest–Shamir–Adleman | RSA Algorithm Explained | Simplilearn

How does RSA Cryptography work?

Figure 3: Study Page 2



Encryption

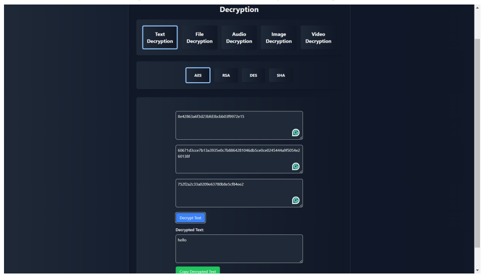
hello

Encrypted Text:
1de49b4c832a24ff753a3ea10b698096

Key:
d755070792c0f82f1538c1e19e572d9a5c7c028ea2a8e5c0133956add1552684

IV:
101c57e38476322a38233a089288d397

Figure 4: Encryption Page 1



Decryption

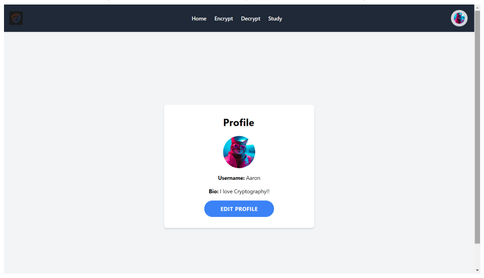
8e42863a6f3d23bfd3bcbb03f9972e15

60671d3cce7b13a3935e0c7b8864281046db5ce0ce0245444a9f5054e260138f

752f2a2c33a0209e63780b8e5cf84ee2

Decrypted Text:
hello

Figure 5: Decrypton Page



Profile

Username: Aaron

Bio: I love Cryptography!!

Figure 6: Profile Page

4.2 Cryptography Engine Design

The cryptography engine is the heart of the system, responsible for executing cryptographic processes such as key generation, encryption, and decryption. Key generation involves creating secure cryptographic keys that can be used for both symmetric and asymmetric encryption. Encryption algorithms, such as AES for symmetric encryption and RSA for asymmetric encryption, are implemented to convert plain text into cipher text securely. Decryption algorithms perform the inverse operation, converting cipher text back into plain text. The engine is designed to be flexible, supporting multiple cryptographic algorithms and allowing users to select their preferred methods. This flexibility is crucial for educational purposes, as it enables users to experiment with and understand different cryptographic techniques.

4.3 User Interface Design

The user interface (UI) is designed to be intuitive and accessible, facilitating user interaction with the cryptography engine. For a command-line interface (CLI), the UI design focuses on simplicity and clarity, with well-defined commands and options that guide users through various cryptographic operations. Each command is designed to be straightforward, ensuring that users can easily perform tasks such as generating keys, encrypting data, and decrypting data without extensive prior knowledge. If a graphical user interface (GUI) is implemented, it will feature a more visual approach, with buttons, menus, and dialogues that make the cryptographic processes more tangible and easier to understand. The GUI design aims to provide a more interactive and engaging experience, particularly for users who prefer visual interaction over text-based commands. Both UI designs emphasize user-friendliness, ensuring that the simulation can serve as an effective educational tool.

5 Algorithms

The Cryptographic simulation uses AES, DES,RSA,SHA Algorithms for key generation.

- **AES (Advanced Encryption Standard)**: It is a symmetric encryption algorithm widely used in cybersecurity for securing data. It encrypts and decrypts data using a single key, ensuring confidentiality. Known for its efficiency and strong security, AES is used in various applications, including secure communications, file encryption, and VPNs.
- **RSA (Rivest-Shamir-Adleman)**: It is an asymmetric encryption algorithm crucial in cybersecurity, utilizing a pair of public and private keys for encryption and decryption. It enables secure data transmission by encrypting data with the public key, which can only be decrypted by the corresponding private key. RSA is commonly used for securing sensitive data, digital signatures, and key exchanges.
- **DES (Data Encryption Standard)**: It is an outdated symmetric encryption algorithm that encrypts and decrypts data using a single key. Once widely used, it has largely been replaced due to vulnerabilities and limited key size (56 bits), making it susceptible to brute-force attacks. DES played a foundational role in cryptographic development, leading to more secure algorithms like AES.
- **SHA (Secure Hash Algorithm)**: It is not used for encryption/decryption but for generating a fixed-size hash value from input data, ensuring data integrity. Common versions like SHA-256 produce unique hashes for different inputs, making it useful for verifying data integrity and authenticity. SHA algorithms are widely used in digital signatures, password hashing, and blockchain technology.

5.1 Tools and Libraries Used

- **OpenSSL Library:** A robust, full-featured open-source toolkit implementing the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. OpenSSL provides a comprehensive suite of cryptographic algorithms, making it a crucial component for handling cryptographic operations in our project.
- **C Programming Language:** Known for its performance and control over system resources, C is used for developing the core cryptographic engine. Its efficiency and

low-level capabilities make it suitable for implementing encryption, decryption, and key generation algorithms.
- **Integrated Development Environment (IDE):** An IDE named Visual Studio Code is used to streamline the development process, providing tools for writing, testing, and debugging code. These IDEs support C programming and integrate well with version control systems, enhancing collaboration and productivity.
- **Version Control System (Git):** Git is used to manage the project's source code, track changes, and facilitate collaboration among team members. It ensures that code can be safely modified and maintained, with a history of changes that can be referenced or reverted if necessary.

6 Implementation

The implementation phase of the cryptography simulation project focuses on setting up the development environment, utilizing essential tools and libraries, and coding the cryptographic algorithms. The development environment is configured with necessary software tools including a C compiler, integrated development environment (IDE), and version control system (e.g., Git) to streamline the coding and collaboration process. The OpenSSL library is integrated into the project to handle cryptographic functions. This library provides robust, industry-standard cryptographic operations, ensuring the project's security and reliability. OpenSSL offers a comprehensive suite of cryptographic functions widely used in various applications.

The core of the implementation involves coding various cryptographic algorithms. For symmetric encryption, algorithms like *Advanced Encryption Standard (AES)* and *Data Encryption Standard (DES)* are implemented to secure data by converting plain text into cipher text using a shared secret key. Asymmetric encryption algorithms, such as *RSA (Rivest-Shamir-Adleman)* is included to provide secure key exchange and digital signatures by utilizing a pair of public and private keys. Additionally, hash functions like *SHA-256* and MD5 are implemented to generate unique, fixed-size hash values from input data, which are essential for data integrity verification and password hashing. By incorporating these diverse cryptographic techniques, the project offers a comprehensive platform for users to explore and understand different aspects of cryptography.

6.1 Features

The cryptographic simulation includes the following features to enhance user interaction and learning:

- **Profile Page:** Allows users to create and manage their profiles, including username, bio, profile picture, and editing profile anytime.
- **Home Page:** Serves as the main dashboard, providing an overview of available cryptographic tools, recent activity, and shortcuts to key features.
- **Sign in/Sign up Page:** There is a sign up and sign in page for users to create individual accounts.
- **Study Page:** An interactive area where users can explore and learn about different cryptographic algorithms. This includes detailed explanations, visual aids, and examples.

- **Encryption and Decryption Page:** A dedicated section for users to encrypt and decrypt data using various algorithms. Users can input plain-text or cipher text and select the desired algorithm for the operation. We have video, audio, file, and image encryption/decryption areas provided.

7 Result & Discussion

The cryptographic simulation project delivers an interactive platform where users can engage with various cryptographic algorithms through a user-friendly interface. The core features of the platform include encryption, decryption, and key generation using well-known algorithms such as AES, RSA, DES, and SHA. The project successfully integrates the OpenSSL library for handling cryptographic operations, ensuring robust and secure processes.

8 Conclusions and Scope for Future Work

In conclusion, this project successfully develops an interactive simulation using OpenSSL in C, aimed at enhancing users' understanding and practical knowledge of cryptographic techniques. By integrating a robust cryptographic engine with an intuitive user interface and comprehensive documentation, the project provides a comprehensive platform for exploring encryption, decryption, key generation, and various cryptographic algorithms. This tool significantly advances digital security education by offering a practical, hands-on learning environment, thus contributing to users' proficiency in cryptographic processes and overall digital security competence.

Future works could focus on expanding the range of supported cryptographic algorithms and incorporating advanced features such as real-time threat detection and response simulations. Additionally, integrating the simulation with cloud-based platforms could facilitate collaborative learning and remote experimentation. Another potential direction is developing mobile applications to make the interactive learning tool more accessible. Continuous updates and improvements based on user feedback and advancements in cryptographic research will ensure the tool remains relevant and effective in educating future users about digital security.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to participate in this Internship. First and foremost, we would like to extend our sincere thanks to our team mentor Siju Swamy Sir for his invaluable guidance and constant support throughout the project. We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources, and sessions on Cryptography. We extend our gratitude to all the researchers, scholars, and experts in the field of Cryptography,Key Generation and Cryptographic Algorithm, whose seminal work has paved the way for our project.

References

[1] JAMGEKAR, R. S., AND JOSHI, G. S. File encryption and decryption using secure RSA. *International Journal of Emerging Science and Engineering (IJESE) 1*, 4 (2013), 11–14.

[2] KÜSTERS, R., AND TUENGERTHAL, M. Ideal key derivation and encryption in simulation-based security. In *CryptographersâĂŹ Track at the RSA Conference* (2011), Springer, pp. 161–179.

[3] NAMBIAR, V. P., KHALIL-HANI, M., AND ZABIDI, M. M. Accelerating the AES encryption function in OpenSSL for embedded systems. *International Journal of Information and Communication Technology 2*, 1-2 (2009), 83–93.

[4] SHARMA, R., DANGI, S., AND MISHRA, P. A comprehensive review on encryption based open source cyber security tools. In *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)* (2021), IEEE, pp. 614–619.

[5] YOUNG, A., AND YUNG, M. An elliptic curve asymmetric backdoor in Open-SSL RSA key generation, 2005.