CRYPTOGUARD FILE PROTECTOR

A Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

R.DILEEP REDDY (2010030416)

K.DIMPLE (2010030436)

K.SIRISHA (2010030438)

M.ROSHAN (2010030441)

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
K L DEEMED TO BE UNIVERSITY
AZIZNAGAR, MOINABAD , HYDERABAD-500 075
MARCH 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **CRYPTOGUARD FILE PROTECTOR** is a bonafide record of the work done by

R.DILEEP REDDY (2010030416)

K.DIMPLE (2010030436)

K.SIRISHA (2010030438)

M.ROSHAN(2010030441)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD , HYDERABAD-500 075**, during the year 2023-2024.

Dr.S.BALAJI
Project Guide

Dr.ARPITA GUPTA
Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

The CryptoGuard File Protector project provides a comprehensive solution for securing digital assets in today's cyber-threat landscape. Its core feature is robust encryption using state-of-the-art cryptographic algorithms like AES, ensuring files remain inaccessible to unauthorized parties. Additionally, sophisticated access controls based on user authentication and role-based permissions regulate file access.

Moreover, CryptoGuard incorporates features for verifying file integrity, detecting unauthorized modifications, and tampering attempts through file integrity checks and digital signatures. Its user-friendly interface facilitates easy deployment, management of security settings, and monitoring file activity via a centralized dashboard.

Cross-platform compatibility ensures support for various operating systems and file formats, allowing users to protect their data consistently across Windows, macOS, Linux, and mobile platforms. Overall, CryptoGuard File Protector empowers individuals and organizations to safeguard their valuable assets from unauthorized access and cyber threats through encryption, access control, integrity checks, and an intuitive interface.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Dr.S.BALAJI, our project guide, for helping us and guiding us in the course of this project .

Dr.ARPITA GUPTA, the Head of the Department, Department of DEPARTMENT NAME.

Our internal reviewers, **Dr. S. BALAJI**, **Mr. D. VINOD KUMAR**, **Dr. P. LALITHA KUMARI** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Background of the Project

The CryptoGuard File Protector project was developed in response to the growing cybersecurity challenges faced by individuals and organizations worldwide. Its primary objective is to provide a comprehensive solution for effectively securing digital assets in today's digital era.

At its core, CryptoGuard focuses on implementing robust encryption using advanced cryptographic algorithms like AES, ensuring that files are inaccessible to unauthorized parties and minimizing the risk of data breaches.

In addition to encryption, the project incorporates sophisticated access control mechanisms based on user authentication and role-based access control. This allows administrators to define precise permissions, dictating who can access, modify, or share protected files, thus enhancing overall security.

Furthermore, CryptoGuard includes features for verifying file integrity through robust checks and digital signatures, providing assurance of data authenticity and trustworthiness.

The project emphasizes a user-friendly interface, simplifying deployment and management of security settings. Users can easily encrypt files, manage access permissions, and monitor file activity via a centralized dashboard.

1.1.1 Cryptography Foundation

The Cryptography Foundation serves as a cornerstone in modern cybersecurity, providing the fundamental principles and techniques necessary to secure digital communications and protect sensitive data. Its primary focus lies in the development and implementation of cryptographic algorithms and protocols, which form the basis for ensuring confidentiality, integrity, and authenticity in information exchange. By leveraging mathematical principles and encryption methodologies, the Cryptography Foundation enables individuals, organizations, and governments to establish secure communication channels, safeguard critical infrastructure, and mitigate the risks posed by cyber threats. Additionally, the foundation plays a pivotal role in advancing cryptographic research, fostering innovation, and promoting best practices to address the evolving challenges of cybersecurity in an increasingly interconnected world.

Secure Verification

In CryptoGuard File Protector, secure verification plays a pivotal role in ensuring the integrity and authenticity of protected files. Through advanced verification mechanisms, the system enables users to authenticate the validity of files and detect any unauthorized alterations or tampering attempts. This process begins with the implementation of robust file integrity checks, which involve computing cryptographic checksums or hashes for protected files. These checksums serve as unique digital fingerprints, represent- ing the contents of each file in a condensed format. During verification, CryptoGuard recalculates the checksums of protected files and compares them against the original values. Any discrepancies indicate potential tampering, triggering alerts and preventing unauthorized access to compromised files.

Moreover, CryptoGuard incorporates digital signatures as an additional layer of verification, enabling users to verify the authenticity and origin of files. Digital signatures are generated using cryptographic techniques, where a private key is used to sign the contents of a file, producing a unique signature. This signature can be verified using the corresponding public key, confirming the integrity of the file and the identity of the

signer. By integrating these secure verification mechanisms, CryptoGuard File Protector empowers users to maintain the integrity of their digital assets, detect unauthorized modifications, and ensure the authenticity of shared files, thereby enhancing overall data security and trustworthiness.

1.2 Problem Statement

The problem statement for the CryptoGuard File Protector project revolves around the escalating cybersecurity threats faced by individuals and organizations in today's digital landscape. With the proliferation of cyberattacks, data breaches, and unauthorized access incidents, there is an urgent need for a comprehensive solution to safeguard sensitive digital assets effectively. Existing file protection methods often fall short in providing robust encryption, access control, and integrity verification mechanisms, leaving valuable data vulnerable to exploitation and compromise. Additionally, the lack of user-friendly interfaces and cross-platform compatibility further exacerbates the challenge of securing files across diverse environments. Thus, there is a critical need for an advanced file protection solution like CryptoGuard that addresses these shortcomings and offersa holistic approach to ensuring the confidentiality, integrity, and authenticity of digital files.

1.3 Objectives

Enhance Data Confidentiality: Implement robust encryption mechanisms, such as AES, to ensure that sensitive files remain inaccessible to unauthorized parties, thereby safeguarding the confidentiality of digital assets.

Strengthen Access Control: Integrate sophisticated access control mechanisms, including user authentication and role-based permissions, to regulate file access and mitigate the risk of unauthorized viewing, modification, or sharing.

Ensure Data Integrity: Incorporate advanced features for verifying the integrity of protected files through file integrity checks and digital signatures, enabling users to detect and prevent unauthorized modifications or tampering attempts.

Provide User-Friendly Interface: Develop an intuitive and user-friendly interface that simplifies the deployment, configuration, and management of security settings, facilitating seamless file protection and monitoring for users of all levels of expertise.

1.4 Scope of the Project

The scope of the CryptoGuard File Protector project encompasses a comprehensive solution for securing digital assets in today's cybersecurity landscape. At its core, the project aims to address the pressing need for robust file protection mechanisms to ensure the confidentiality, integrity, and authenticity of sensitive data. This includes implementing advanced encryption algorithms, such as AES, to render files inaccessible to unauthorized parties and mitigate the risk of data breaches. Additionally, the project focuses on integrating sophisticated access control mechanisms, including user authentication and role-based permissions, to regulate file access and prevent unauthorized viewing, modification, or sharing. Moreover, CryptoGuard emphasizes the importance of verifying the integrity of protected files through file integrity checks and digital signatures, enabling users to detect and prevent tampering attempts effectively. Furthermore, the project seeks to provide a user-friendly interface and ensure cross-platform compatibility to facilitate seamless deployment and management of security settings across diverse user environments. Overall, the scope of the CryptoGuard File Protector project encompasses a holistic approach to file protection, aiming to empower individuals and organizations to safeguard their valuable digital assets from cyber threats effectively.

Chapter 2

Literature Review

2.1 Overview on Literature

Analyze encryption and public key infrastructure (PKI): Encryption is like a secret code for your data, making it unreadable to unauthorized people. Public Key Infrastructure (PKI) is like a digital key management system that helps protect your secrets. In PKI, you have a pair of keys: a public key (for locking) and a private key (for unlocking). When someone wants to send you a secret message, they use your public key to lock it, and only your private key can unlock and reveal the message. This technology ensures secure communication and keeps your data safe from prying eyes. This paper says in the cloud platforms, TLS with PKI is a better solution because it allows cryptographically secured methods of authentication in environments that are unsafe.

Security evaluation of diferent hashing functions with rsa for digital signature: To evaluate the security aspects of various hashing functions in conjunction with RSA for digital signature applications. The review examines the strengths and weaknesses of different hashing algorithms, such as SHA-256, SHA-3, and MD5, concerning collision resistance, pre-image resistance, and computational efficiency. Furthermore, it explores how these hashing functions complement RSA, a widely adopted asymmetric encryption algorithm, in generating secure digital signatures. The review scrutinizes existing research findings and methodologies employed in assessing the security properties of hashing functions when integrated with RSA for digital signature schemes. Additionally, it highlights recent advancements and emerging trends in the field, shedding light on potential avenues for future research and development efforts aimed at enhancing

the security and efficiency of digital signature systems.

Enhanced Digital Signature Framework for Web Applications: The architecture for digital signatures in web applications offers a structured framework enabling usersto securely sign documents or data online, thereby ensuring content authenticity and integrity. This framework typically comprises several key components: a user-friendly interface facilitating the signing process, cryptographic algorithms for generating digital signatures, a robust Public Key Infrastructure (PKI) for key management, and a verification mechanism to validate signatures. Digital signatures play a crucial role in establishing trust and thwarting tampering attempts in digital documents, making them indispensable for secure online transactions, contracts, and communication within web applications. This architecture ensures the reliability and security of digital signatures, bolstering confidence in online interactions and transactions.

2.2 Example on File Protector Scenario

Alice works for a financial institution and frequently deals with sensitive client information stored in digital files. She needs to ensure that these files are protected from unauthorized access or tampering, especially when sharing them with colleagues or partners. To address this, Alice decides to use CryptoGuard File Protector, a comprehensive solution for securing digital assets.

Using CryptoGuard, Alice encrypts the sensitive files containing client data using robust encryption algorithms like AES. This ensures that even if unauthorized parties gain access to the files, they cannot decipher the contents without the appropriate decryption key. Additionally, she sets up access controls within CryptoGuard, specifying who can view, modify, or share the protected files based on user roles and permissions.

Furthermore, Alice enables file integrity checks and digital signatures within CryptoGuard to verify the authenticity and integrity of the protected files. This ensures that any unauthorized modifications or tampering attempts are detected, preventing potential data breaches or manipulation of sensitive information.

With CryptoGuard's user-friendly interface, Alice can easily manage the security

settings and monitor file activity through a centralized dashboard. Whether she's working on her desktop computer or accessing files from her mobile device, CryptoGuard offers seamless cross-platform compatibility, ensuring consistent data protection across different environments.

Overall, CryptoGuard File Protector empowers Alice to safeguard sensitive client information effectively, providing peace of mind knowing that her digital assets are secure from unauthorized access or tampering, even when shared with colleagues or partners.

2.3 Developed Technology on Cryptography in Digital Signatures

In recent years, significant advancements have been made in the technology of cryptography, particularly in the realm of digital signatures. Traditional digital signature schemes relied heavily on asymmetric encryption algorithms like RSA and cryptographic hash functions like SHA-256. However, emerging technologies have expanded the landscape of digital signatures, offering more robust security and efficiency. One notable development is the adoption of elliptic curve cryptography (ECC) for digital signatures, which offers smaller key sizes while maintaining strong security properties. ECC-based signature schemes, such as ECDSA (Elliptic Curve Digital Signature Algorithm), have gained popularity due to their ability to provide the same level of security as traditional algorithms but with reduced computational overhead and key sizes. Additionally, advancements in post-quantum cryptography have introduced new signature schemes resistant to attacks from quantum computers, ensuring the long-term security of digital signatures in an era of evolving threats. Moreover, innovations in multi-party computation (MPC) and homomorphic encryption have enabled secure and privacy-preserving digital signature schemes, allowing parties to collaborate on signing documents without revealing sensitive information. Overall, these developments represent significant progress in the technology of cryptography for digital signatures, offering enhanced security, efficiency, and privacy in various applications, including

secure transactions, document authentication, and identity verification.

2.4 Overview of Related Works

In the realm of cryptography, a vast body of related works spans various subfields, each contributing to the development and enhancement of cryptographic techniques and applications. Research in symmetric-key cryptography focuses on designing algorithms such as AES and DES, aiming to ensure efficient and secure encryption and decryption processes. Similarly, asymmetric cryptography research delves into the complexities of algorithms like RSA and ECC, striving to provide robust mechanisms for key exchange and digital signatures. Furthermore, cryptographic hash functions like SHA-256 and MD5 are extensively studied to ensure collision resistance and data integrity verification. Within the realm of cryptographic protocols, works explore the design and analysis of secure communication protocols like SSL/TLS and SSH, which are essential for securing data transmission over networks. Additionally, advancements in post-quantum cryptography aim to develop algorithms resistant to quantum attacks, paving the way for future-proof encryption solutions. Overall, this diverse array of related works in cryptography collectively contributes to strengthening the security and privacy of digital systems and communication channels in our increasingly interconnected world.

2.5 Advantages and Limitations of existing systems

The existing system for the CryptoGuard File Protector project offers several advantages, but it also has some limitations:

Advantages:

Robust Encryption: The system utilizes robust encryption algorithms like AES to ensure the confidentiality of protected files, making it highly secure against unauthorized access.

Sophisticated Access Control: It incorporates sophisticated access control mechanisms, including user authentication and role-based permissions, allowing administrators to regulate file access effectively.

Integrity Verification: The system includes features for verifying the integrity of protected files through file integrity checks and digital signatures, enabling users to detect and prevent tampering attempts.

User-Friendly Interface: With a user-friendly interface, the system facilitates easy deployment and management of security settings, making it accessible to users of all levels of expertise.

Cross-Platform Compatibility: It ensures compatibility with various operating systems and file formats, enabling consistent data protection across different environments.

Limitations:

Performance Overhead: The encryption and integrity verification processes may impose a performance overhead, particularly on systems with large files or high processing demands.

Complexity for Administrators: Setting up and managing access controls and security settings may be complex for administrators, requiring a significant level of expertise.

Dependency on Cryptographic Standards: The system's security relies on the strength and reliability of cryptographic algorithms and standards, making it vulnerable to weaknesses or vulnerabilities in these components.

Cost of Implementation: Implementing and maintaining the system, particularly in large-scale deployments, may incur significant costs in terms of infrastructure, training, and ongoing support.

Potential Compatibility Issues: While the system aims for cross-platform compatibility, compatibility issues with certain operating systems or file formats may still arise, limiting its effectiveness in certain environments.

Overall, while the existing system offers robust security features and usability, addressing its limitations may require ongoing refinement and adaptation to meet evolving cybersecurity challenges and user needs.

Chapter 3

Proposed System

3.1 System Requirements

By defining clear and comprehensive system requirements, the CryptoGuard File Protector project can ensure the successful development and deployment of a robust and user-friendly solution for securing digital assets.

Operating System Compatibility: The software should be compatible with major operating systems such as Windows, macOS, and Linux, ensuring broad usability across different environments.

Hardware Requirements: Define the minimum hardware specifications required for running the software efficiently, considering factors such as processor speed, memory (RAM), and disk space.

Encryption Algorithms: Specify the supported encryption algorithms, such as AES, and ensure compatibility with industry standards to guarantee secure data protection.

Access Control Mechanisms: Implement user authentication and role-based access control mechanisms, allowing administrators to manage user permissions effectively.

File Integrity Checks: Include features for verifying the integrity of protected files through cryptographic checksums or digital signatures, ensuring data integrity.

User Interface: Design a user-friendly interface that simplifies the encryption process, access control management, and monitoring of file activity, enhancing usability for both administrators and end-users.

Cross-Platform Compatibility: Ensure compatibility with various web browsers

and mobile platforms to enable seamless access and management of protected files across different devices.

Scalability: Design the system to scale effectively to accommodate growing numbers of users, files, and transactions without compromising performance or security.

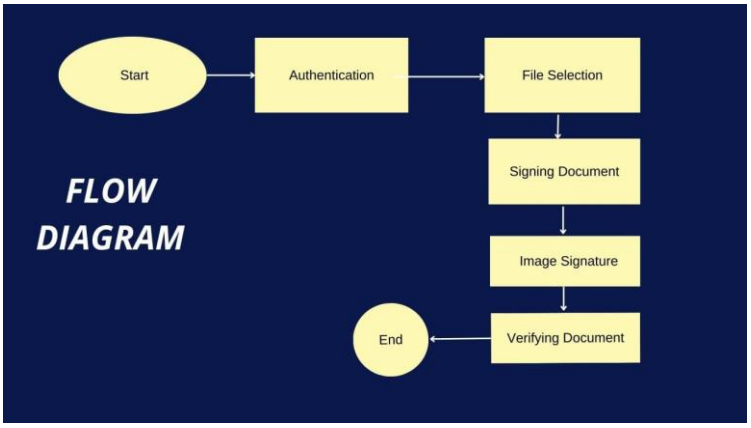
Integration with PKI: Integrate with a Public Key Infrastructure (PKI) for managing cryptographic keys, certificate issuance, and revocation, enhancing the security and trustworthiness of digital signatures.

Logging and Auditing: Implement logging and auditing mechanisms to track user activities, file access, and security events, facilitating compliance with regulatory requirements and enabling forensic analysis in the event of security incidents.

Documentation and Support: Provide comprehensive documentation, user guides, and support resources to assist users in deploying, configuring, and troubleshooting the system effectively.

3.1.1 Flow Chart of the System

The system design centers on creating an intuitive user interface for document signing and verification, coupled with robust cryptographic algorithms to ensure data security. It emphasizes efficient key management, document integrity, and compliance with legal standards.



3.2 Design of the System

The design of the CryptoGuard File Protector system encompasses a structured framework aimed at providing robust security features while ensuring usability and compatibility across different environments. At its core, the system incorporates a modular architecture comprising several key components. These include encryption modules utilizing industry-standard algorithms like AES to ensure data confidentiality, access control mechanisms facilitating user authentication and role-based permissions to regulate file access, and file integrity verification modules employing cryptographic checksums or digital signatures to detect and prevent tampering attempts. Additionally, the system features a user-friendly interface facilitating seamless deployment and management of security settings, along with cross-platform compatibility to ensure consistent data protection across various operating systems and file formats. The design emphasizes scalability to accommodate growing user needs and transactions while maintaining optimal performance and security standards. Overall, the CryptoGuard File Protector system is meticulously designed to provide comprehensive file protection solutions, empowering users to safeguard their digital assets effectively in today's cybersecurity landscape.ator pitch that focuses on what and why without delving into how.

3.2.1 Encryption and Decryption Keys

In the CryptoGuard File Protector project, encryption and decryption keys play a pivotal role in ensuring the security and confidentiality of protected files. Encryption keys are utilized during the encryption process to transform plaintext data into ciphertext, rendering it unintelligible to unauthorized parties. These keys are generated using cryptographic algorithms such as AES and are typically kept secret to prevent unauthorized access to the encrypted data. Decryption keys, on the other hand, are used during the decryption process to revert ciphertext back to its original plaintext form. In asymmetric encryption schemes like RSA, decryption keys are distinct from encryption keys and are typically generated as a key pair consisting of a public key for encryption and a private key for decryption. The private key must be kept confidential to ensure the security

of the decryption process, while the public key can be freely distributed for encryption purposes. In the context of CryptoGuard, the system securely manages both encryption and decryption keys, employing robust key management practices to safeguard sensitive cryptographic material and ensure the integrity and confidentiality of protected files.

3.3 Algorithms and Techniques used

The CryptoGuard File Protector project employs a range of algorithms and techniques to ensure the security and integrity of protected files. Some of the key algorithms and techniques used in the project include:

Encryption Algorithms: The project utilizes symmetric encryption algorithms such as AES (Advanced Encryption Standard) to encrypt the contents of files. AES is a widely adopted encryption algorithm known for its security and efficiency in protecting data.

Hash Functions: Cryptographic hash functions like SHA-256 are used for generating checksums or digital signatures to verify the integrity of protected files. These hash functions ensure that any modifications to the file can be detected by comparing the computed hash with the original value.

Public Key Infrastructure (PKI): The project leverages PKI for managing cryptographic keys, including key generation, distribution, and revocation. PKI facilitates secure communication and authentication between users and ensures the confidentiality and integrity of encryption keys.

Access Control Mechanisms: Techniques such as user authentication and role-based access control (RBAC) are employed to regulate file access based on user permissions and privileges. These mechanisms ensure that only authorized individuals have the necessary privileges to access, modify, or share protected files.

Digital Signatures: Digital signatures generated using asymmetric encryption algorithms like RSA are used to authenticate the origin and integrity of files. Digital signatures provide assurance that the file has not been tampered with and that it originates from a trusted source.

Cross-Platform Compatibility: Techniques are implemented to ensure compatibility with various operating systems and file formats, allowing users to protect their data consistently across different environments.

Overall, by incorporating these algorithms and techniques, the CryptoGuard File Protector project aims to provide a comprehensive solution for securing digital assets and protecting them from unauthorized access or tampering.

3.3.1 SHA-256 Algorithm

In the CryptoGuard File Protector project, the SHA-256 algorithm is utilized as a cryptographic hash function for generating checksums or digital signatures to ensure the integrity of protected files. SHA-256, which belongs to the SHA-2 (Secure Hash Algorithm 2) family, produces a 256-bit hash value from an input message of arbitrary length. This hash value serves as a unique digital fingerprint of the input data, making it virtually impossible for two different sets of data to produce the same hash value.

In the context of CryptoGuard, SHA-256 is employed for various purposes: **File Integrity Verification:** When a file is encrypted or protected, its SHA-256 hash value is computed and stored alongside the encrypted data. Upon decryption or access, CryptoGuard recalculates the SHA-256 hash of the decrypted file and compares it with the stored hash value to ensure that the file has not been tampered with.

Digital Signatures: In some cases, SHA-256 may be used in conjunction with asymmetric encryption algorithms like RSA to generate digital signatures for files. The file's SHA-256 hash value is signed with the sender's private key, providing authentication and ensuring the integrity of the file during transmission or sharing.

By incorporating SHA-256 into the CryptoGuard File Protector project, the sys- tem can effectively verify the integrity of protected files and provide assurance against unauthorized modifications or tampering attempts. This enhances the overall security and trustworthiness of the protected data, making it suitable for sensitive applications such as secure communication, document management, and data storage.

Chapter 4

Implementation

4.1 Tools and Technologies used

The CryptoGuard File Protector project utilizes a variety of tools and technologies to implement its functionalities effectively. Some of the key tools and technologies used in the project include:

Programming Languages: Languages such as React, html, css may be used for implementing the core functionalities of the CryptoGuard software, including encryption algorithms, access control mechanisms, and user interface components.

Cryptography Libraries: Cryptographic libraries such as OpenSSL or Bouncy Castle provide a wide range of cryptographic functions and algorithms needed for encryption, decryption, hashing, digital signatures, and key management within the CryptoGuard system.

User Interface Frameworks: Frameworks like Qt (for desktop applications) or React (for web applications) may be utilized to develop the user interface of the CryptoGuard application, providing a visually appealing and user-friendly experience for interacting with the software.

Public Key Infrastructure (PKI) Solutions: PKI solutions such as OpenSSL or Microsoft Active Directory Certificate Services (AD CS) may be employed to manage cryptographic keys, issue digital certificates, and facilitate secure communication and authentication within the CryptoGuard system.

Version Control Systems: Version control systems like Git are used for managing the source code of the CryptoGuard project, enabling collaboration among developers,

tracking changes, and ensuring code integrity and versioning control.

Integrated Development Environments (IDEs): IDEs such as Visual Studio Code, PyCharm, or Eclipse provide a development environment with features such as code editing, debugging, and project management, facilitating the development and testing of the CryptoGuard software.

Cross-Platform Development Tools: Tools like Electron or PyQt allow developers to create cross-platform applications compatible with multiple operating systems, ensuring that the CryptoGuard software can be deployed on Windows, macOS, and Linux environments.

Documentation Tools: Documentation tools like Sphinx or Doxygen may be used to generate comprehensive documentation for the CryptoGuard project, including user guides, API documentation, and developer documentation, ensuring clarity and accessibility for users and contributors.

By leveraging these tools and technologies, the CryptoGuard File Protector project can efficiently develop, deploy, and maintain a robust and user-friendly solution for securing digital assets and protecting sensitive data from unauthorized access or tampering.

4.1.1 Architecture of the Project

The architecture of the CryptoGuard File Protector project encompasses a modular and layered design to ensure robust security features and scalability. At its core, the architecture comprises several key components, including encryption modules responsible for implementing industry-standard algorithms like AES for file encryption and decryption. These modules interact with access control mechanisms that regulate file access based on user permissions and privileges, ensuring that only authorized individuals can view, modify, or share protected files. Additionally, cryptographic hash functions such as SHA-256 are utilized for generating checksums or digital signatures to verify the integrity of protected files. The architecture also incorporates a user-friendly interface layer, facilitating seamless deployment and management of security settings through a

centralized dashboard. Moreover, the architecture emphasizes cross-platform compatibility to ensure consistent data protection across diverse operating systems and environments. Overall, the modular and layered architecture of CryptoGuard enables efficient development, deployment, and maintenance of a comprehensive solution for securing digital assets and protecting sensitive data from unauthorized access or tampering.

4.2 Modules and their descriptions

Signing involves digitally marking a document with a unique cryptographic signature, and it also involves the images of the signature which cannot be tampered by others and the system is ensuring its integrity and source authenticity. Verification is the process of confirming the signature's validity, ensuring the document hasn't been tampered with and was signed by the claimed source. Both processes are essential for secure digital transactions and document management.

4.2.1 Signatures and Verification

Signature: It applies a unique cryptographic signature to a document, ensuring its authenticity and preventing unauthorized modifications. This module is responsible for 20 digitally marking a document with a unique cryptographic signature. It ensures the integrity of the document and verifies its source authenticity. By employing robust cryptographic algorithms, this module safeguards the document against tampering and unauthorized access. The signing process is fundamental for establishing the trustworthiness of digital documents in various applications, such as contracts, agreements, and digital records.

Verification: The verification module plays a crucial role in confirming the validity of digital signatures. It checks whether the document has been tampered with and ensures that it was indeed signed by the claimed source. Verification is essential for maintaining the security and trustworthiness of digital transactions and document management, as it helps in detecting any unauthorized alterations to the signed documents. It checks the signature to confirm the document's integrity and that it was indeed signed

by the claimed source, maintaining trust in digital transactions.

4.3 Flow of the System

The system works as you pick a document, then you can choose to add a digital signature to it or you can check if a signature on a document is real. To make sure everything is safe, the system takes care of special codes called keys, and it keeps your documents and signatures safe. If something goes wrong or right, the system tells you, and it makes sure everything follows the rules. So, it's like having a special seal on your digital papers, and a super-smart system to check it.

4.3.1 Cryptography Key Benefits

Cryptography offers several key benefits in the realm of information security:

Confidentiality: One of the primary benefits of cryptography is confidentiality. It allows for the encryption of sensitive data, ensuring that only authorized parties with the appropriate decryption keys can access the information. This protects data from unauthorized access, eavesdropping, and interception by malicious actors.

Integrity: Cryptography helps ensure the integrity of data by providing mechanisms to detect any unauthorized modifications or tampering attempts. Hash functions and digital signatures enable the verification of data integrity, ensuring that the information has not been altered during transmission or storage.

Authentication: Cryptography enables authentication, allowing parties to verify the identity of each other in digital communications. Techniques such as digital signatures and public key infrastructure (PKI) enable the authentication of messages and entities, ensuring trustworthiness in online transactions and communications.

Non-Repudiation: Cryptography supports non-repudiation, which means that a party cannot deny the authenticity or validity of a digital signature or transaction. Digital signatures provide cryptographic proof of the origin and integrity of messages or transactions, preventing parties from disowning their actions.

Secure Communication: Cryptography enables secure communication over inse-

cure channels, such as the internet. By encrypting data and using secure communication protocols like SSL/TLS, cryptography ensures that information exchanged between par- ties remains confidential and protected from eavesdropping or interception.

Data Protection: Cryptography helps protect data at rest and in transit, safeguarding it from unauthorized access, theft, or tampering. Encryption techniques can be applied to files, databases, emails, and other forms of digital communication, ensuring that sensitive information remains secure even if it falls into the wrong hands.

Compliance: Cryptography plays a crucial role in compliance with various regulations and standards related to data protection and privacy, such as GDPR, HIPAA, and PCI DSS. Implementing cryptographic measures helps organizations meet regulatory requirements and avoid potential legal and financial consequences associated with data breaches.

Overall, cryptography provides essential tools and techniques for ensuring the confidentiality, integrity, authenticity, and non-repudiation of digital information, making it indispensable in modern information security practices.

Chapter 5

Results and Analysis

5.1 Performance Evaluation

Performance evaluation helps identify areas for improvement, ensures the system meets its intended objectives, and provides confidence in its reliability and security.

Security Assessment: Conduct vulnerability assessments and penetration testing to identify potential security flaws.

User Experience: Collect user feedback on the system's ease of use, user interface, and overall experience.

Processing Speed: Measuring the time required for signing and verifying documents of various sizes. This evaluates the efficiency of cryptographic operations, ensuring documents can be processed swiftly.

Resource Utilization: Monitoring CPU and memory usage during document signing and verification. This is essential to guarantee that the system operates efficiently without consuming excessive resources.

Scalability: Evaluating how the system performs as the number of signed documents and users increase. Scalability ensures that the system can handle growing demands without a significant drop in performance.

Compliance: Ensuring the system complies with relevant data security and digital signature standards and regulations. Compliance is essential for legal validity and trust.

Error Handling: Testing the system's ability to handle errors, such as incorrect document formats or failed verification attempts. Effective error handling enhances the system's reliability.

Scalability: Evaluate how the system performs as the number of signed documents and users increases.

5.1.1 Authentication of Cryptography

Authentication in cryptography refers to the process of verifying the identity of communicating parties and ensuring the integrity of transmitted data. This critical aspect of cryptography is essential for establishing trust and security in digital communications and transactions. One common method of authentication is through the use of digital signatures, which provide cryptographic proof of the origin and integrity of messages or documents. Digital signatures are created using asymmetric encryption algorithms, such as RSA or ECDSA, where a signer uses their private key to encrypt a hash of the message, resulting in a unique signature. The recipient can then use the signer's public key to verify the signature, confirming both the authenticity of the sender and the integrity of the message. Another authentication technique is through the use of cryptographic authentication protocols, such as HMAC (Hash-based Message Authentication Code), which use cryptographic hash functions to verify the integrity of transmitted data. These protocols generate a unique tag or code based on the data and a secret key, which can be verified by the recipient to ensure the authenticity and integrity of the message. Overall, authentication is a fundamental aspect of cryptography that enables secure and trustworthy communication, ensuring that parties can verify each other's identities and the integrity of transmitted data.

5.2 Comparison with existing systems

In comparison with existing systems, the CryptoGuard File Protector project offers several distinct advantages and improvements in securing digital assets and protecting sensitive data. Unlike some existing systems that may rely solely on basic encryption techniques, CryptoGuard incorporates robust encryption algorithms such as AES, providing a higher level of security against unauthorized access and data breaches. Additionally, CryptoGuard integrates sophisticated access control mechanisms, including

user authentication and role-based permissions, allowing administrators to regulate file access more effectively and granularly. Furthermore, CryptoGuard emphasizes the importance of verifying the integrity of protected files through features like file integrity checks and digital signatures, offering enhanced assurance against tampering attempts and data manipulation. Moreover, the user-friendly interface and cross-platform compatibility of CryptoGuard make it more accessible and convenient for users to deploy and manage security settings across diverse environments. Overall, the CryptoGuard File Protector project represents a significant advancement in file protection solutions, offering comprehensive security features and usability improvements compared to existing systems.

5.2.1 AES in Cryptography

AES is widely regarded as a cornerstone of modern encryption technology, playing a crucial role in securing data in various applications, including communication, storage, and authentication. Its combination of security, efficiency, and versatility makes it a preferred choice for protecting sensitive information in today's digital world.

5.3 Limitations and future scope

While the CryptoGuard File Protector project offers significant advancements in file protection and security, it also has certain limitations and areas for future improvement:

Limitations:

Performance Overhead: The encryption and decryption processes may imposea performance overhead, particularly on systems with large files or high processing demands. Optimizing these processes to minimize latency and resource consumption could enhance the overall performance of the system.

Complexity for Administrators: Setting up and managing access controls and security settings may be complex for administrators, requiring a significant level of expertise. Simplifying the user interface and providing more intuitive management tools could improve usability for administrators.

Dependency on Cryptographic Standards: The security of the system relies on the strength and reliability of cryptographic algorithms and standards. Any weaknesses or vulnerabilities in these components could potentially undermine the security of the system.

Future Scope:

Enhanced Performance: Future developments could focus on optimizing encryption and decryption algorithms to improve performance without compromising security. This could involve leveraging hardware acceleration, parallel processing techniques, or implementing more efficient cryptographic primitives.

Usability Improvements: Further enhancements to the user interface could make the system more user-friendly and accessible to a wider range of users. Providing guided setup wizards, contextual help, and comprehensive documentation could stream- line the deployment and management processes.

Integration with Emerging Technologies: Integration with emerging technologies such as blockchain or homomorphic encryption could enhance the security and functionality of the system. Exploring new cryptographic techniques and protocols could also address evolving threats and security challenges.

Scalability and Compatibility: Enhancements to scalability and compatibility could ensure that the system can accommodate growing numbers of users, files, and transactions while remaining compatible with diverse operating systems and environments.

Continuous Security Updates: Regular security updates and patches could help address vulnerabilities and ensure that the system remains resilient against emerging threats. Implementing a robust security update mechanism could help protect users' data from evolving security risks.

By addressing these limitations and exploring future opportunities, the CryptoGuard File Protector project can continue to evolve and provide effective solutions for securing digital assets in an increasingly interconnected and digitized world.

5.3.1 Realtime Monitoring

Real-time monitoring capabilities in the CryptoGuard File Protector project provide crucial insights into file activity and security events, enhancing the overall security posture of the system. By implementing real-time monitoring, administrators can actively track and analyze file access, modifications, and other security-related events as they occur. This enables prompt detection and response to potential security threats, such as unauthorized access attempts or suspicious file activity. Real-time monitoring features may include alerts and notifications triggered by predefined security events, allowing administrators to take immediate action to mitigate risks and protect sensitive data. Additionally, comprehensive logging and auditing functionalities enable administrators to review historical file activity and security events, facilitating forensic analysis and compliance with regulatory requirements. Overall, real-time monitoring in the CryptoGuard File Protector project empowers administrators to proactively monitor and manage the security of digital assets, ensuring timely detection and response to security incidents in dynamic and evolving environments.

Chapter 6

Conclusion and Recommendations

6.1 Summary of the Project

The CryptoGuard File Protector project is a comprehensive solution designed to address the critical need for securing digital assets in today's cybersecurity landscape. At its core, the project employs robust encryption mechanisms, including industry-standard algorithms like AES, to ensure the confidentiality of protected files. Additionally, sophisticated access control mechanisms, such as user authentication and role-based permissions, regulate file access and prevent unauthorized viewing, modification, or sharing. The system also incorporates features for verifying the integrity of protected files through file integrity checks and digital signatures, providing assurance against tampering attempts.

The user-friendly interface of CryptoGuard facilitates easy deployment and management of security settings, while cross-platform compatibility ensures consistent data protection across various operating systems and file formats. Real-time monitoring capabilities empower administrators to actively track file activity and security events, enabling prompt detection and response to potential threats. Moreover, future enhancements may include optimizations for performance, usability improvements, integration with emerging technologies, and continuous security updates to address evolving threats. Overall, the CryptoGuard File Protector project represents a proactive approach to securing digital assets, empowering individuals and organizations to safeguard their sensitive data effectively in today's dynamic and interconnected digital environment.

6.2 Contributions and achievements

The contributions and achievements of the CryptoGuard File Protector project are significant in the realm of cybersecurity and data protection. Some key contributions and achievements include:

Enhanced Data Security: The project significantly enhances data security by providing a comprehensive solution for encrypting and protecting digital assets. By employing robust encryption algorithms and access control mechanisms, CryptoGuard ensures the confidentiality and integrity of sensitive files, safeguarding them from unauthorized access and tampering.

User-Friendly Interface: The project's user-friendly interface makes it accessible to users of all levels of expertise, facilitating seamless deployment and management of security settings. This usability improvement enhances the adoption and effective- ness of the solution among individuals and organizations seeking to protect their digitalassets.

Cross-Platform Compatibility: CryptoGuard's cross-platform compatibility ensures consistent data protection across various operating systems and file formats, offering flexibility and convenience to users working in diverse environments.

Real-Time Monitoring and Response: The inclusion of real-time monitoring capabilities empowers administrators to actively track file activity and security events, enabling prompt detection and response to potential threats. This proactive approach enhances the overall security posture of users' digital assets.

Continuous Improvement and Adaptation: The project's commitment to continuous improvement and adaptation ensures that it remains resilient against emerging threats and evolving cybersecurity challenges. Future enhancements, such as performance optimizations, usability improvements, and integration with emerging technologies, further strengthen its effectiveness and relevance in securing digital assets.

Overall, the CryptoGuard File Protector project's contributions and achievements play a crucial role in empowering individuals and organizations to protect their sensitive data effectively in today's dynamic and interconnected digital landscape.

6.3 Recommendations for future work

For future work, several recommendations could further enhance the effectiveness and scope of the CryptoGuard File Protector project. Firstly, exploring advancements in encryption algorithms and cryptographic techniques could bolster the system's security posture. Research into post-quantum cryptography and homomorphic encryption may offer innovative approaches to protect data against emerging threats. Additionally, enhancing the system's scalability and performance through optimization techniques and leveraging cloud computing resources could accommodate larger datasets and improve overall efficiency.

Furthermore, integrating blockchain technology could introduce decentralized and tamper-resistant data storage solutions, enhancing data integrity and auditability. Embracing artificial intelligence and machine learning algorithms for anomaly detection and threat prediction could also fortify the system's ability to detect and respond to security incidents in real-time. Moreover, fostering collaboration with cybersecurity researchers and industry experts to conduct comprehensive security assessments and penetration testing would help identify and address potential vulnerabilities proactively.

User-centric improvements such as developing mobile applications for on-the-go file protection and implementing multi-factor authentication mechanisms could enhance user experience and strengthen access controls. Lastly, staying abreast of regulatory developments and compliance requirements to ensure alignment with evolving data protection regulations would be essential. By incorporating these recommendations, the CryptoGuard File Protector project can continue to evolve as a leading-edge solution for safeguarding digital assets in an ever-changing cybersecurity landscape.

Bibliography

[1] HTML 4.01 Specification, World Wide Web Consortium (W3C) specification, Dec. 1999; www.w3.org/ TR/html401.

[2] S. Mazumdar, "XML Digital Signature Tool," Mozilla Add-Ons, 2012; https://addons.mozilla.org/en-Us/ thunderbird/addon/xml-digital-signature-tool.

[3] A. Rossanagel. "Digital Signature Regulation and European Trends." http://www.emr-sb.de/news/Dsregulation.pdf

[4] CBrenn. Summary of the Australian Law on Electronic signatures. http://rechten.kub.n.lsimone/bretm.htm

[5] Digital Signature Trust. Certainsend Security: A Brief Technical Overview, http:l/www.trustdst.com/prod$_s$*crv/certainsendtech – overview*

Appendices

Appendix A

Source code

App.py

```
from flask import Flask, render_template, request, redirect, url_for, flash
import os
from werkzeug.utils import secure_filename
from PIL import Image
import hashlib

app = Flask(__name__)
app.secret_key = "your_secret_key"
app.config['UPLOAD_FOLDER'] = 'uploads'

@app.route('/about')
def about():
    return render_template('about.html')


def calculate_hash(file_path):
    """Calculate hash of a file."""
    sha256_hash = hashlib.sha256()
    with open(file_path, "rb") as f:
        for byte_block in iter(lambda: f.read(4096), b""):
            sha256_hash.update(byte_block)
    return sha256_hash.digest()


def encrypt_hash_with_image(hash_bytes, image_path):
    """Encrypt hash with image."""
    with open(image_path, 'rb') as img_file:
        img_bytes = img_file.read()
    encrypted_bytes = bytes(h1 ^ h2 for h1, h2 in zip(hash_bytes, img_bytes))
    return encrypted_bytes


def decrypt_hash_with_image(encrypted_bytes, image_path):
    """Decrypt hash using image."""
    with open(image_path, 'rb') as img_file:
        img_bytes = img_file.read()
```

```
    decrypted_bytes = bytes(h1 ^ h2 for h1, h2 in zip(encrypted_bytes,
img_bytes))
    return decrypted_bytes


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/sign', methods=['GET', 'POST'])
def sign():
    if request.method == 'POST':
        document = request.files['document']
        image = request.files['image']

        # Ensure the 'uploads' directory exists
        if not os.path.exists(app.config['UPLOAD_FOLDER']):
            os.makedirs(app.config['UPLOAD_FOLDER'])

        # Save the document and image
        document_filename = secure_filename(document.filename)
        document_path = os.path.join(app.config['UPLOAD_FOLDER'],
document_filename)
        document.save(document_path)
        image_filename = secure_filename(image.filename)
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], image_filename)
        image.save(image_path)

        # Calculate document hash
        document_hash = calculate_hash(document_path)

        # Encrypt document hash with image
        encrypted_hash = encrypt_hash_with_image(document_hash, image_path)

        # Save encrypted hash as signature
        signature_filename = document_filename + '.sig'
        signature_path = os.path.join(app.config['UPLOAD_FOLDER'],
signature_filename)
        with open(signature_path, 'wb') as sig_file:
            sig_file.write(encrypted_hash)

        flash('Document signed successfully!', 'success')
        return redirect(url_for('index'))
    return render_template('sign.html')


@app.route('/verify', methods=['GET', 'POST'])
def verify():
    if request.method == 'POST':
```

```
        provided_document = request.files['document']
        provided_image = request.files['image']

        # Save the provided document and image
        provided_document_filename =
secure_filename(provided_document.filename)
        provided_document_path = os.path.join(app.config['UPLOAD_FOLDER'],
provided_document_filename)
        provided_document.save(provided_document_path)
        provided_image_filename = secure_filename(provided_image.filename)
        provided_image_path = os.path.join(app.config['UPLOAD_FOLDER'],
provided_image_filename)
        provided_image.save(provided_image_path)

        # Calculate hash of provided document
        provided_document_hash = calculate_hash(provided_document_path)

        # Decrypt hash with provided image
        signature_filename = provided_document_filename + '.sig'
        signature_path = os.path.join(app.config['UPLOAD_FOLDER'],
signature_filename)
        encrypted_hash = open(signature_path, 'rb').read()
        decrypted_hash = decrypt_hash_with_image(encrypted_hash,
provided_image_path)

        # Compare hashes
        if decrypted_hash == provided_document_hash:
            flash('Document verified successfully!', 'success')
        else:
            flash('Document verification failed!', 'error')

        # Delete temporary files
        os.remove(provided_document_path)
        os.remove(provided_image_path)

        return redirect(url_for('index'))

    return render_template('verify.html')


if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Crypto Guard File Protector</title>
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='styles.css') }}"
    />

  </head>
  <body style="background-color: rgb(220, 220, 220)">
  <body>

    <header>
      <h1>Crypto Guard File Protector</h1>
      <ul>
        <li><a href="{{ url_for('index') }}">Home</a></li>

        <li><a href="{{ url_for('about') }}">About</a></li>

        <li><a href="{{ url_for('sign') }}">Sign Document</a></li>

        <li><a href="{{ url_for('verify') }}">Verify Document</a></li>
      </ul>
    </header>

    <main>
      <h3>
        This application allows users to digitally sign and verify documents
        using images as passwords.

      </h3>
      <h4>Developed By:</h4>
      <h4>2010030416 - Dileepreddy</h4>
      <h4>2010030436 - Dimple</h4>
      <h4>2010030416 - Sirisha</h4>
      <h4>2010030416 - Roshan</h4>
    </main>
  </body>
</html>
```

sign.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Sign Document</title>
  </head>
  <body style="background-color: rgb(220, 220, 220)"></body>
  <body>
    <h1>Sign Document</h1>
    <style>
      h1 {
        text-align: center;
      }
      p {
        text-align: center;
      }
      div {
        text-align: center;
      }
      h1 {
        color: #0620a1;
      }
    </style>
    <form method="POST" enctype="multipart/form-data">
      <label for="document">Select Document:</label><br />
      <input
        type="file"
        id="document"
        name="document"
        accept=".pdf"
      /><br /><br />
      <label for="image">Select Image for Signature:</label><br />
      <input type="file" id="image" name="image" accept="image/*" /><br /><br
/>
      <button type="submit">Sign</button>
    </form>

    <!-- Display selected image -->
    <img
      id="selectedImage"
      src="#"
      alt="Selected Image"
      style="display: none; max-width: 300px"
    />

    <script>
      // Function to display selected image
      function displayImage(input) {
```

```
        if (input.files && input.files[0]) {
          var reader = new FileReader();
          reader.onload = function (e) {
            document.getElementById("selectedImage").src = e.target.result;
            document.getElementById("selectedImage").style.display = "block";
          };
          reader.readAsDataURL(input.files[0]);
        }
      }

      // Call displayImage function when a new image is selected
      document.getElementById("image").addEventListener("change", function ()
{
        displayImage(this);
      });
    </script>
  </body>
</html>
```

verify.html

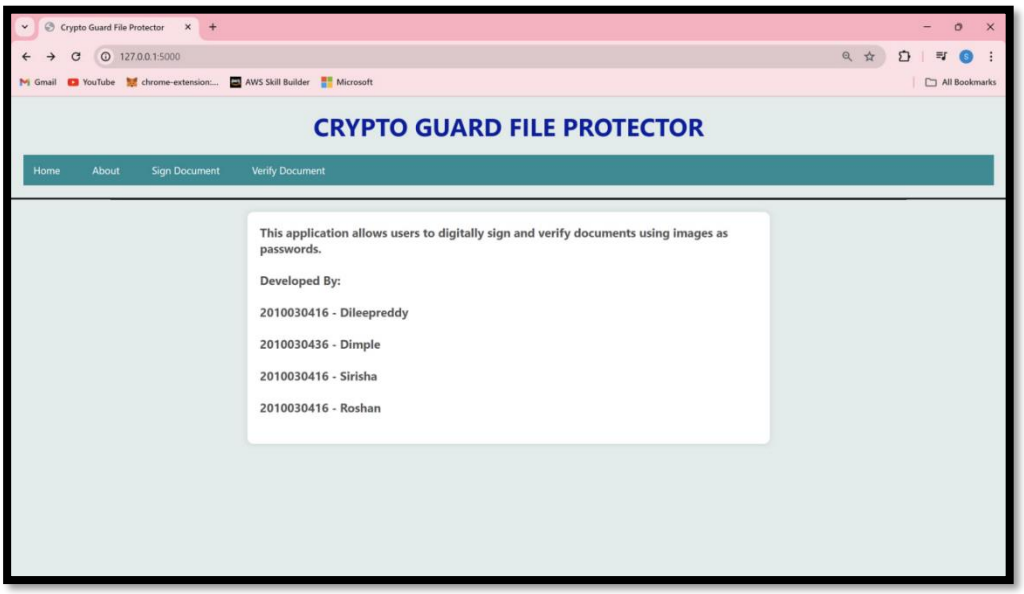
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='styles.css') }}"
    />
    <title>Verify Document</title>
  </head>
  <body style="background-color: rgb(220, 220, 220)"></body>
  <body>
    <h1>Verify Document</h1>
    {% with messages = get_flashed_messages() %} {% if messages %}
    <ul>
      {% for message in messages %}
      <li>{{ message }}</li>
      {% endfor %}
    </ul>
    {% endif %} {% endwith %}
    <form method="POST" enctype="multipart/form-data">
      <label for="document">Select Document:</label><br />
      <input
        type="file"
        id="document"
        name="document"
        accept=".pdf"
      /><br /><br />
```

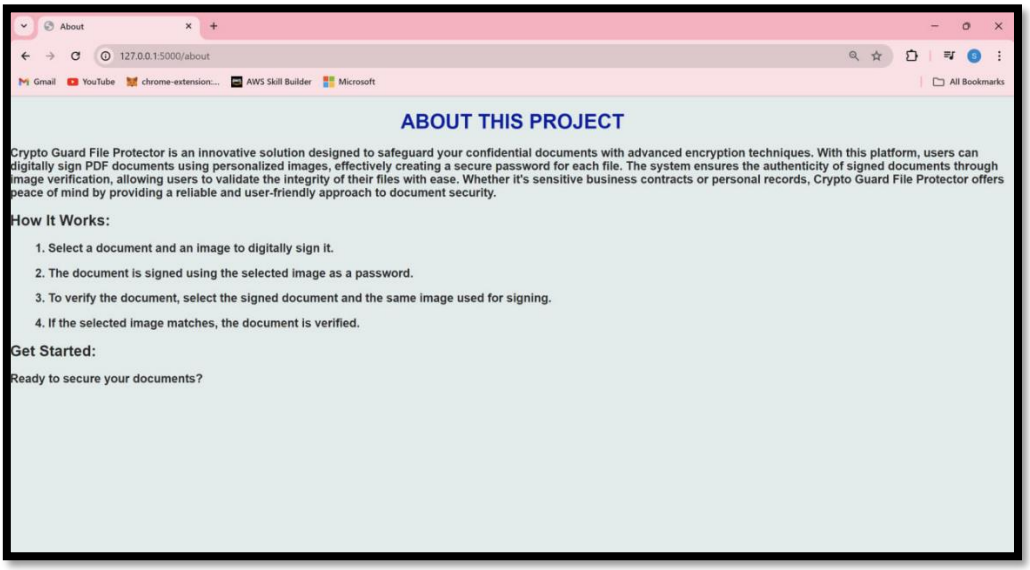
```
      <label for="image">Select Image for Verification:</label><br />
      <input type="file" id="image" name="image" accept="image/*" /><br />
      <span id="imageFilename"></span><br /><br />
      <!-- Display selected image filename here -->
      <button type="submit">Verify</button>
    </form>

    <!-- Display selected image -->
    <img
      id="selectedImage"
      src="#"
      alt="Selected Image"
      style="display: none; max-width: 300px"
    />

    <script>
      // Function to display selected image
      function displayImage(input) {
        if (input.files && input.files[0]) {
          var reader = new FileReader();
          reader.onload = function (e) {
            document.getElementById("selectedImage").src = e.target.result;
            document.getElementById("selectedImage").style.display = "block";
          };
          reader.readAsDataURL(input.files[0]);
        }
      }

      // Call displayImage function when a new image is selected
      document.getElementById("image").addEventListener("change", function ()
{
        displayImage(this);
      });
    </script>

    <script>
      // Update image filename when an image is selected
      document.getElementById("image").addEventListener("change", function ()
{
        document.getElementById("imageFilename").innerText =
this.files[0].name;
      });
    </script>
  </body>
</html>
```

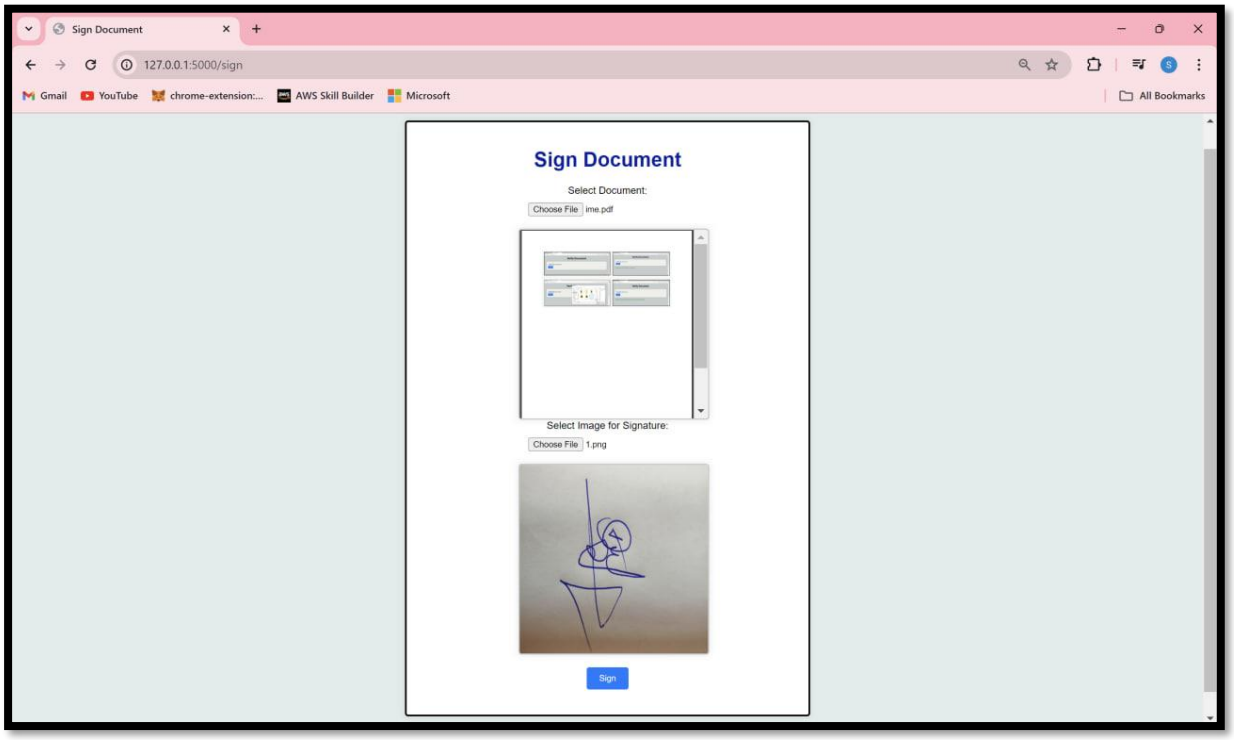
Appendix B

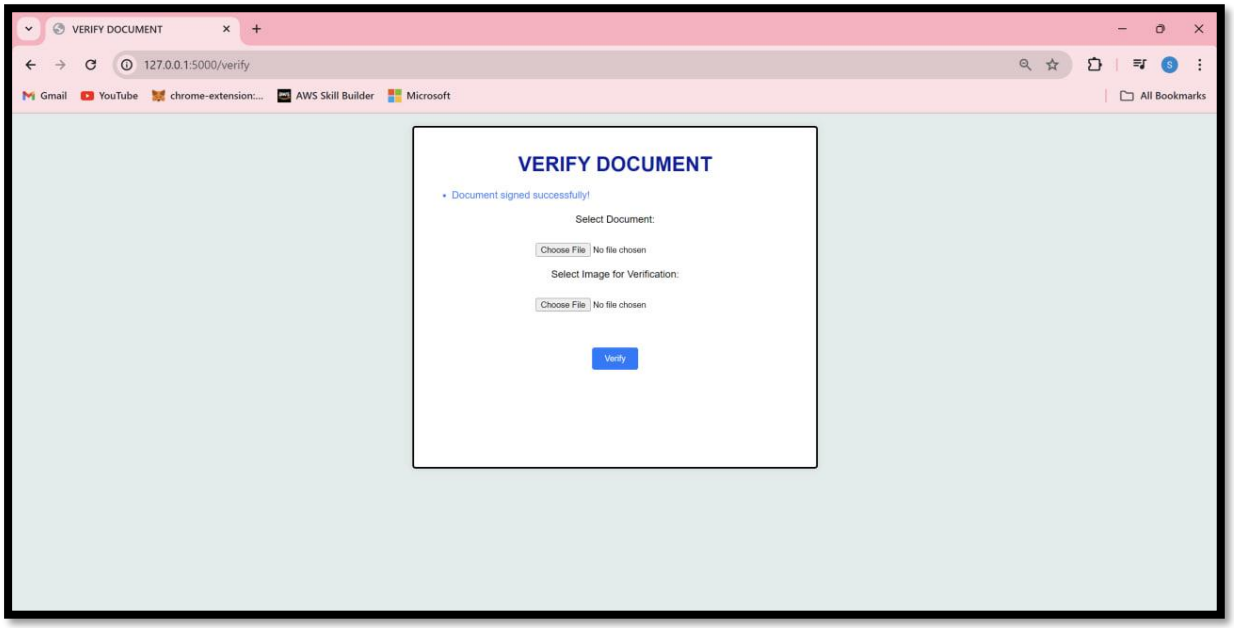
Screen shots

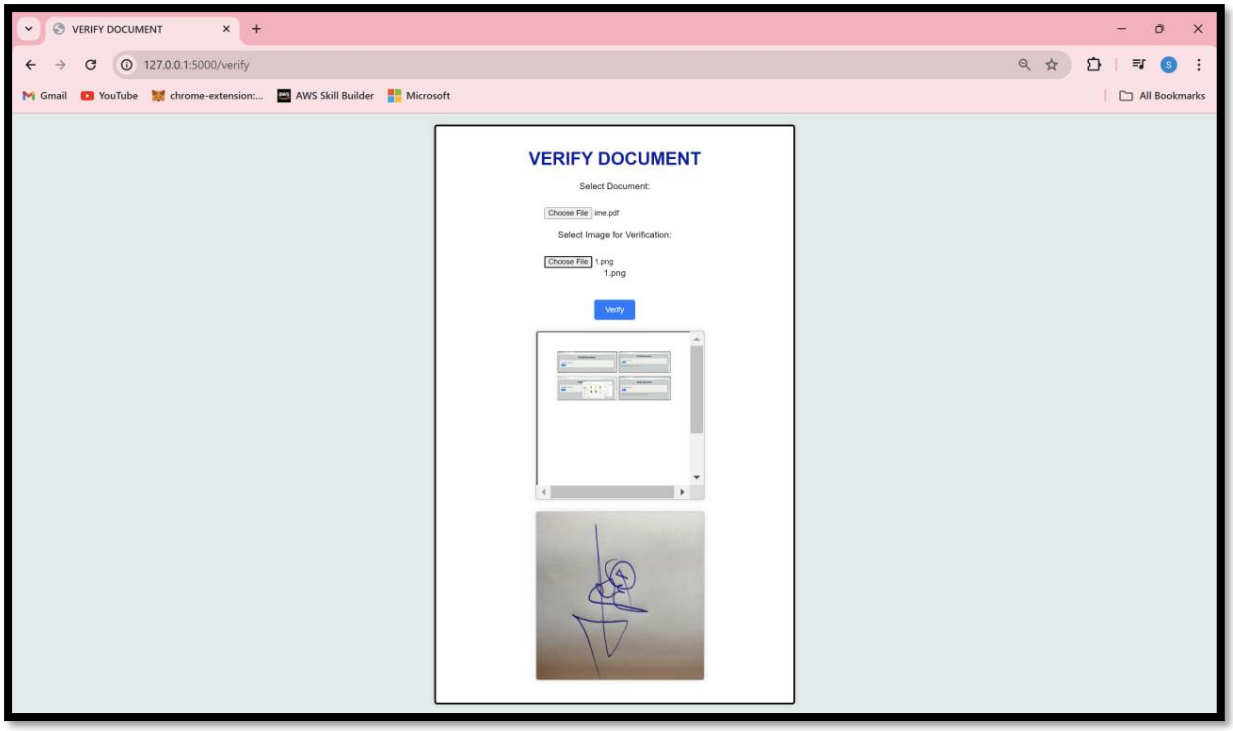
B.1 Fixed Image

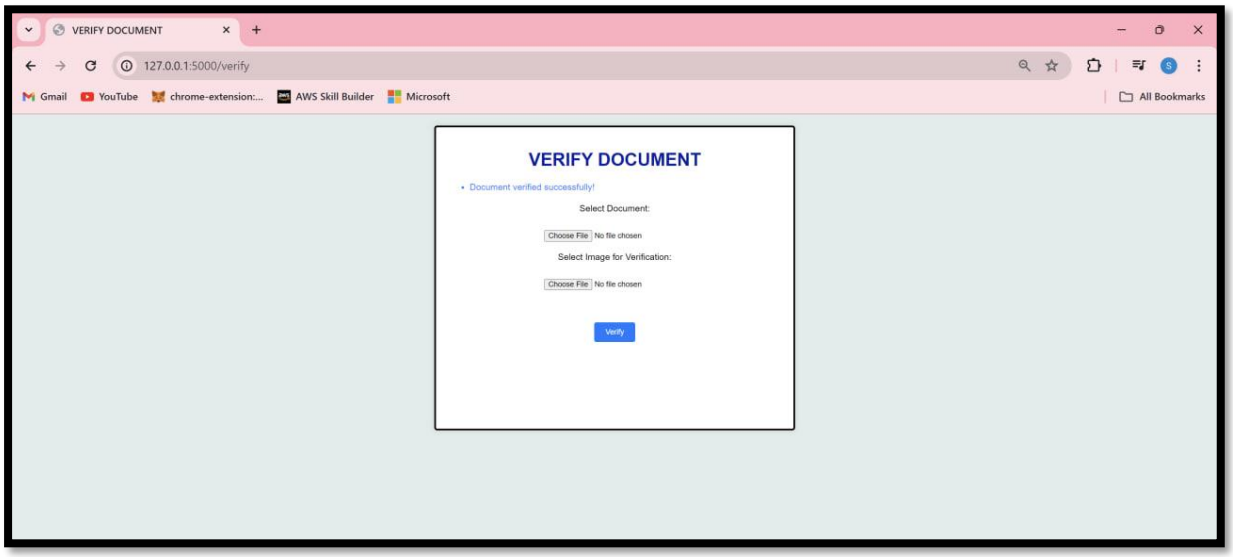


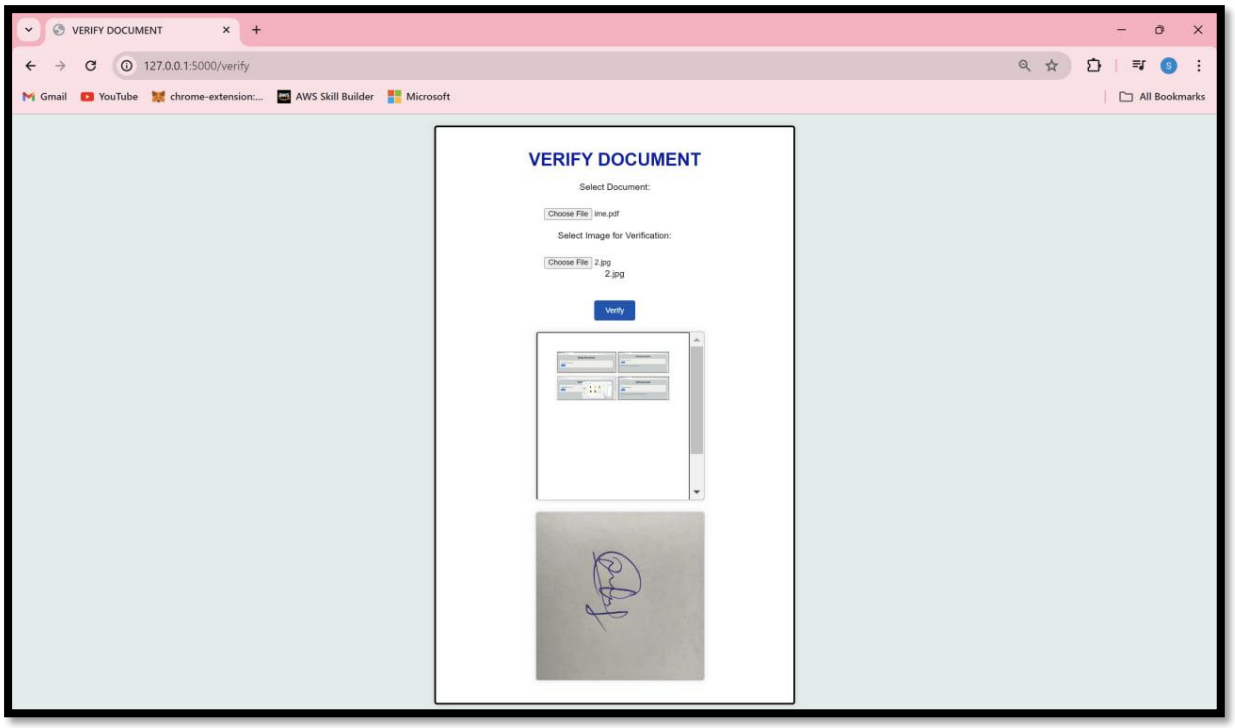


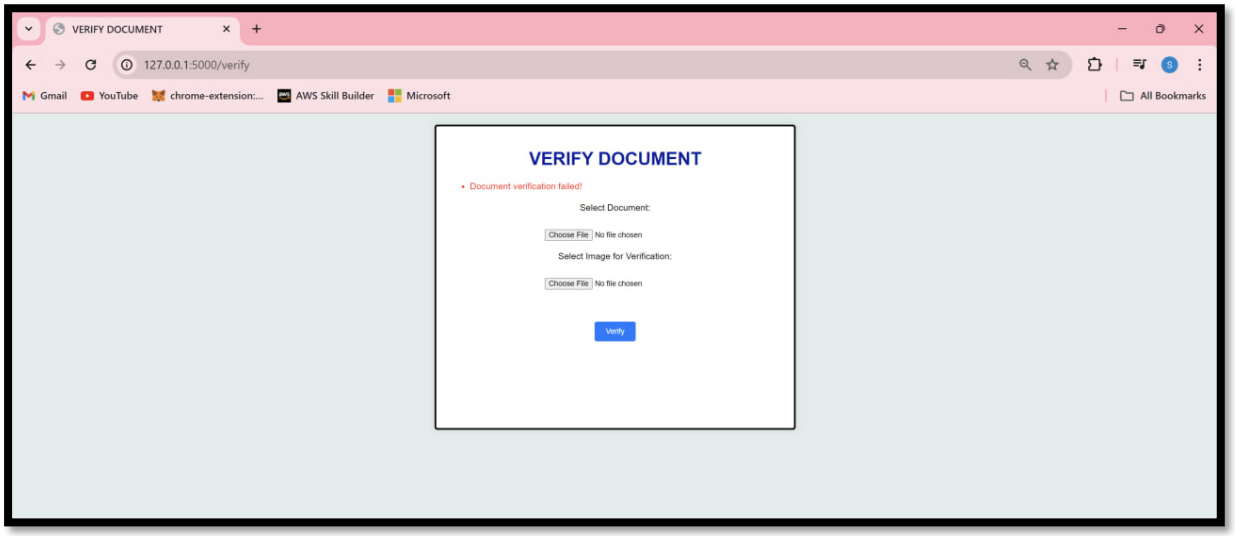

Sign Document
127.0.0.1:5000/sign
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
Sign Document
Select Document:
Choose File
ime.pdf
Select Image for Signature:
Choose File
1.png
Sign


VERIFY DOCUMENT
127.0.0.1:5000/verify
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
VERIFY DOCUMENT
Document signed successfully!
Select Document:
Choose File
No file chosen
Select Image for Verification:
Choose File
No file chosen
Verify


VERIFY DOCUMENT
127.0.0.1:5000/verify
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
VERIFY DOCUMENT
Select Document:
Choose File
ime.pdf
Select Image for Verification:
Choose File
1.png
1.png
Verify


VERIFY DOCUMENT
127.0.0.1:5000/verify
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
VERIFY DOCUMENT
Document verified successfully!
Select Document:
Choose File
No file chosen
Select Image for Verification:
Choose File
No file chosen
Verify


VERIFY DOCUMENT
127.0.0.1:5000/verify
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
VERIFY DOCUMENT
Select Document:
Choose File
Select Image for Verification:
Choose File
2.jpg
Verify


VERIFY DOCUMENT
127.0.0.1:5000/verify
Gmail
YouTube
chrome-extension:...
AWS Skill Builder
Microsoft
All Bookmarks
VERIFY DOCUMENT
Document verification failed!
Select Document:
Choose File
No file chosen
Select Image for Verification:
Choose File
No file chosen
Verify

Appendix C

Dataset used in the project