TITLE: COMPREHENSIVE REPORT OUTLINE: DIGITAL SIGNATURE VERIFICATION SECURITY SYSTEM

NAME: CLIFFORD KINGSLEY BAIDOO

INDEX NUMBER: FOE.41.018.037.22

INSTRUCTOR’S NAME: MS. AUDREY ASANTE

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

INSTITUTION: UNIVERSITY OF MINES AND TECHNOLOGY, TARKWA

DATE OF SUBMISSION: 04/03/2025

TABLE OF CONTENTS

Contents

Abstract

In an age where digital communication is essential to daily operations, maintaining the integrity and validity of incoming emails and file attachments has become critical. While digital signatures provide a dependable form of verification, most existing solutions rely on centralized infrastructure or manual processes, which restricts accessibility and real-time responses. This project describes the creation of a Signature Verification System that runs locally on a user's PC, automatically checking digital signatures on emails and files as they arrive like how assistive programs like Grammarly monitor text input in real time.

The system detects emails and uploaded files, extracts embedded signatures or hashes, and uses cryptographic algorithms to ensure their integrity. Verified files pass through, whereas modified or unverified stuff is quarantined and logged. The system, built using Python for backend functionality and a React-based dashboard for administrative monitoring, integrates seamlessly with compatible email clients and file explorers, eliminating the need for external mail servers.

1. INTRODUCTION

1.1 BACKGROUND

Digital communication, particularly email, forms the backbone of modern interactions but was designed prioritizing functionality over security. This has created vulnerabilities that are exploited through phishing, spoofing, and malware distribution. Digital signatures emerged as a cryptographic solution using asymmetric key pairs to verify message authenticity and integrity. Standards like S/MIME and PGP established frameworks for email signature implementation, yet many systems lack robust verification mechanisms. This security gap leaves users vulnerable to various attacks, highlighting the need for reliable signature verification at the local level without modifying existing email infrastructure.

1.2 CYBERCURITY PROBLEM

Email remains a primary vector for cyberattacks, with malicious actors exploiting the lack of authentication mechanisms. Without digital signature verification, recipients cannot reliably confirm sender identity or message integrity, leading to successful phishing campaigns, malware distribution, and business email compromise attacks. Organizations face significant risks when accepting unverified files or instructions, as these may contain malicious payloads or fraudulent requests resulting in data breaches, financial losses, or unauthorized system access.

1.3 IMPORTANCE OF DIGITAL SIGNATURE VERIFICATION

Digital signature verification serves as a critical security mechanism that ensures both the authenticity and integrity of communications. By validating that messages originated from the claimed sender and remain unaltered during transmission, signature verification prevents impersonation attacks and message tampering. This technology creates accountability in digital communications, provides legal verification comparable to physical signatures in many jurisdictions, and builds user trust by confirming message legitimacy. In environments handling sensitive information, signature verification becomes essential for regulatory compliance and protection against sophisticated attacks that might otherwise bypass traditional security measures.

1.4 SCOPE

The Sig-Check system focuses exclusively on verifying digital signatures in emails and attachments that have already been received and stored locally on the user's machine. It does not alter email delivery mechanisms or integrate with mail server configurations. The system processes existing email files, validates their digital signatures against trusted certificates, and categorizes messages based on verification results. Sig-Check provides a monitoring interface for verification activities, generates alerts for suspicious content, and maintains comprehensive logs of verification outcomes. This local-first approach offers enhanced security without requiring changes to existing email infrastructure.

1.5 GOALS & OBJECTIVES

- To create a reliable system for verifying digital signatures in locally stored emails
- To detect and flag emails with invalid or missing signatures
- To protect users from potentially malicious attachments through signature validation
- To provide a user-friendly dashboard for monitoring verification activities
- To maintain detailed logs of verification results for security analysis
- To implement the system with minimal impact on existing workflows

2. PROBLEM STATEMENT

2.1 CHALLENGES OF UNAUTHENTICATED TRANSMISSIONS

1. Spoofing and Impersonation: Attackers can impersonate trusted senders, convincing recipients to open malicious attachments or take harmful actions. Without proper signature verification, it becomes difficult to differentiate legitimate communications from fraudulent ones.

2. Tampering of Files: Emails or documents can be intercepted and altered in transit by malicious actors. Without authentication, recipients have no easy way to verify whether the files they receive have been modified, which can lead to the execution of malicious code or the receipt of inaccurate data.
3. Data Breaches: In the absence of secure authentication and verification mechanisms, attackers can exploit the lack of verification to gain unauthorized access to sensitive data, leading to breaches that can have severe legal, financial, and reputational consequences.
4. Loss of Trust: A system that lacks proper digital signature verification diminishes user confidence in digital communications. Users may become hesitant to trust electronic documents, emails, or transactions, especially in environments where confidentiality and integrity are critical.

2.2 JUSTIFICATION

The Sig-Check system addresses a critical vulnerability in conventional email workflows where messages and attachments are often accepted without adequate verification of their source. This gap creates opportunities for various attacks, including malware distribution, phishing, and data exfiltration. By implementing local signature verification, organizations can significantly reduce the risk of compromised systems and data breaches. The financial and reputational costs of security incidents far outweigh the investment in preventative measures like Sig-Check. Furthermore, the local-first approach provides an accessible security enhancement that can be deployed quickly without disrupting existing email infrastructure, making it an ideal solution for organizations seeking to improve their security posture incrementally.

3. PROPOSED SOLUTION

3.1 SIG-CHECK SYSTEM OVERVIEW

Sig-Check operates as a local application that monitors incoming emails, extracts digital signatures, verifies their authenticity, and processes the emails accordingly. Unlike traditional solutions that integrate with mail servers, Sig-Check works with emails that have already been received on the local machine.

3.2 UNIQUE/INNOVATIVE ASPECPTS

- Local-first approach without requiring mail server configuration
- Accepts multiple file types instead of only emails
- Better Automation and Actionable Response
- Integration with other software and infrastructures

3.3 HIGH-LEVEL ARCHITECTURE

The architecture consists of several key components:

1. User's Device (Endpoint): The core of the system is located at the user's device, where incoming emails, PDF documents, and text files are processed. The device runs a background service that listens for new email arrivals or file interactions.
2. Email Client Integration: The system integrates with the user's email client (such as Thunderbird, Outlook, etc.) to detect and validate new emails. It scans email attachments for valid digital signatures and prompts the user with verification results.
3. Signature Verification Service: This component is responsible for validating the digital signatures of incoming documents and emails.
4. File Handler Module: For non-email files such as PDFs and text documents, this module extracts and verifies digital signatures embedded in the files. It uses cryptographic algorithms to ensure data integrity and sender verification.
5. Notification and Reporting System: Once verification is complete, the system notifies the user about the status of the file or email (e.g., whether it is verified, altered, or unverified). In the case of a failed verification, the system can move the file to a quarantine folder and generate an alert.
6. Database: The system stores logs of all verification actions, user interactions, and flagged files. This database serves as a record for future audits and helps maintain transparency in the system’s operations.
7. User Interface (UI): A simple, intuitive interface is provided for users to monitor the verification process, view logs, and access quarantined files.

This architecture ensures that users can independently and efficiently verify the authenticity of digital communications, improving overall cybersecurity and reducing dependence on external infrastructure.

4. SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

Email Processing: Scan and Process emails from local storage

Signature Extraction: Identify and extract digital signatures from emails

Verification Engine: Authenticates signatures from emails

Content Management: Sort verified and unverified emails/attachments

Notification System: Alerts users about verification activities

4.2 NON-FUNCTIONAL REQUIREMENTS

Performance: The system must verify the digital signature of emails and files within 20 seconds

Availability: The system should be highly available since it operates in real-time.

Reliability: The system must be accurate with the signature verification

Usability: The system must provide a user-friendly interface for users to view verification results

4.3 HARDWARE/SOFTWARE REQUIREMENTS

4.3.1 SYSTEM REQUIREMENTS

Operating System: Windows 10 or 11, Linux

Processor: Modern multi-core processor 2.0 GHz or Faster

Memory: 4gb RAM

Storage: 20gb

4.3.2 SOFTWARE DEPENDENCIES

Python: Python 3.8 or higher

Email Client: Thunderbird, Outlook, Gmail

5. IMPLEMENTATION

5.1 TECHNOLOGIES AND TOOLS

5.1.1 PROGRAMMING LANGUAGE

- Python - Used for backend development, implementing signature verification, and handling email processing.
- JavaScript - Used for frontend development, enabling dynamic and interactive elements for the UI
- HTML/CSS - For structuring and styling the frontend of the application, ensuring a user-friendly interface.

5.1.2 FRAMWORK AND LIBRARIES

- React - A JavaScript library for building dynamic user interfaces, allowing the creation of a responsive and interactive dashboard for email monitoring and signature verification.
- Flask - A lightweight web framework for building the backend API, managing routing, and handling logic for email and file processing.

- PyPDF2 - A Python library for handling PDF files, especially for extracting and verifying signatures in PDF documents.
- Cryptography - A library for performing cryptographic functions, such as signature verification, encryption, and decryption.
- Flask-Mail - An extension for Flask used to manage email sending and receiving, facilitating communication between the backend and email services.

5.1.3 DEVELOPMENT TOOLS:

- Ubuntu (WSL) - The operating system used for development, offering a stable and secure environment for building and testing the system.
- Docker - A containerization platform used for creating and managing isolated environments for the application, ensuring consistency across different stages of development, testing, and production.
- VsCode - A code editor used for writing, editing, and debugging both the backend and frontend code, with extensive support for Python, JavaScript, and HTML/CSS.
- Github - A version control and collaboration platform for managing code repositories, tracking changes, and collaborating with team members.

5.1.4 COMPONENT INTEGRATION

The components of the Signature Verification System interact seamlessly to enable real-time digital signature validation.

1. User Interface (Frontend - React)

- Role: The frontend provides the user interface where users can monitor incoming emails, check their status, and initiate the signature verification process.
- Interaction:
  - Users upload email attachments or files on the dashboard.
  - Dashboard to view logs and verification results.
  - React communicates with the Flask API (backend) to send requests for signature verification and display results.

2. Backend (Flask)

- Role: The Flask backend serves as the intermediary between the frontend and the core processing logic. It handles incoming requests from the frontend, processes the data, and returns the results.
- Interaction:

  - It processes the data using the signature verification logic (implemented via Python's Cryptography library).
  - Sends back results, such as success/failure of verification, to be displayed on the frontend.
  - The Flask-Mail extension enables sending notifications (if applicable, such as email verification failures).

3. Signature Verification Logic (Cryptography Library)

- Role: This component is responsible for validating the authenticity and integrity of the digital signature.
- Interaction:
  - Flask routes the data to the verification logic when an email or document is uploaded.
  - The Cryptography library checks if the file's signature matches the expected one, ensuring the data is untampered and originates from the claimed sender.

4. File Handling (PyPDF2 & Attachment Processor)

- Role: This component processes file attachments (such as PDFs or text files) in the received email to extract signatures and other relevant metadata.
- Interaction:
  - Once the file is received and forwarded to the backend, PyPDF2 extracts the digital signature from PDF attachments (if available).
  - If the attachment is another type of file, other libraries or custom code are used to handle those files appropriately.
  - The extracted data is then used for signature verification.

5. Email Handling (Flask-Mail & Email Server)

- Role: This component facilitates the sending and receiving of emails through Flask-Mail. The email server (Postfix) is responsible for routing emails to the system.
- Interaction:

    - Flask-Mail is used to interact with the email server (such as Postfix, if applicable), receiving emails that contain attachments to be processed.
    - When an email arrives at the server, it triggers the backend to fetch the email content and attachments, initiating the verification process.
    - The backend also communicates with the frontend to show whether the email's signature is valid or not.

7. Docker (Containerization)

- Role: Docker provides isolated environments for each service (frontend, backend, email server) to ensure they function consistently across different systems.
- Interaction:
    - Docker containers are used to deploy each part of the application (frontend, backend, email server, etc.), enabling easy setup, scaling, and management.
    - This ensures that each service operates independently but can still communicate with other components (e.g., the Flask backend communicates with the React frontend via HTTP requests).

6. CONCLUSION

This project successfully addresses a critical gap in digital security by introducing real-time, local digital signature verification for emails, PDF documents, and text files. By empowering users to independently verify the authenticity and integrity of received digital content, the system reduces the reliance on centralized infrastructures or manual verification processes. The approach enhances cybersecurity at the user level, mitigating threats such as spoofed emails, tampered attachments, and man-in-the-middle attacks.

Key achievements include the seamless integration of cryptographic verification mechanisms, the development of an intuitive user interface, and the deployment of a lightweight solution that can be easily adopted by users with minimal technical expertise. Despite some limitations, such as support for a limited range of file types and the absence of full email server integration, the system demonstrates considerable potential for improving everyday digital communication security.

The broader impact of this project lies in its potential to shift the responsibility of verification from centralized servers to individual users, thereby promoting a more proactive, decentralized approach to cybersecurity. With future enhancements such as broader file format support and full automation, this solution can become an integral part of personal and organizational security strategies, contributing to the protection of sensitive data and the preservation of digital trust.

Clifford
Kingsley
Baidoo

Digitally signed by
Clifford Kingsley Baidoo
Date: 2025.05.07
09:03:14 Z