DIGITAL SIGNATURE IMPLEMENTATION FOR SECURE TRANSACTIONS

A Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

S.HARISH	2203A51064
MOHD.KARIM PASHA	2203A51054

Under the guidance of

Dr. Santosh Kumar Henge

Associate Professor, School of CS&AI.

SR University,

Ananthasagar, Warangal,Telangana-506371

SR UNIVERSITY

CERTIFICATE

This is to certify that this project entitled **"DIGITAL SIGNATURE IMPLEMENTATION FOR SECURE TRANSACTIONS** " is the bonafied work carried out by **S.HARISH, MOHD.KARIMPASHA** as a Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2024-2025 under our guidance and Supervision.

Dr. Santosh Kumar Henge
Associate Professor
SR University
Ananthasagar,Warangal

Dr. M.Sheshikala
Professor& Head,
School of CS&AI,
SR University
Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Major Project guide **Dr. Santosh Kumar Henge, Associate Professor** as well as Head of the School of CS&AI , **Dr. M.Sheshikala, Professor** and Dean of the School of CS&AI, **Dr.Indrajeet Gupta Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

TABLE OF CONTENTS

1. ABSTRACT

In the era of rapid digital transformation, the security and authenticity of electronic transactions have become critical concerns. Traditional security mechanisms, such as passwords and PINs, are no longer sufficient to protect against evolving cyber threats like phishing, spoofing, and man-in-the-middle attacks. This project focuses on the implementation of digital signatures as a robust solution for securing digital transactions. Digital signatures are based on asymmetric cryptography, utilizing a pair of keys—private and public—to ensure data integrity, authentication, and nonrepudiation.

The objective of this project is to design and implement a system where messages or transaction data are signed by the sender using a private key and verified by the receiver using the corresponding public key. This mechanism guarantees that the data has not been altered in transit and that it originates from a verified sender. The project employs industry-standard algorithms such as RSA and SHA-256 for signature generation and verification. Python, with its cryptography libraries, is used to simulate key generation, signing, and verification processes.

Through a series of experiments and test cases, the system demonstrates that any alteration in the message or use of invalid keys leads to verification failure, thereby validating the effectiveness of digital signatures in real-world applications. The project highlights how digital signatures can enhance the security of online transactions, document exchange, and sensitive communications. It lays a foundation for further research and integration with technologies such as blockchain, secure email systems, and digital certificates for enterprise-level cybersecurity solutions.

2.

INTRODUCTION

In today's digital era, the rapid advancement of technology has revolutionized the way individuals and organizations conduct transactions. From online banking and e-commerce to government services and corporate communications, digital platforms are now the backbone of secure and efficient operations. However, with this digital transformation comes the growing risk of cyber threats, data breaches, identity theft, and fraud. Ensuring the security, authenticity, and integrity of digital transactions has become more critical than ever.

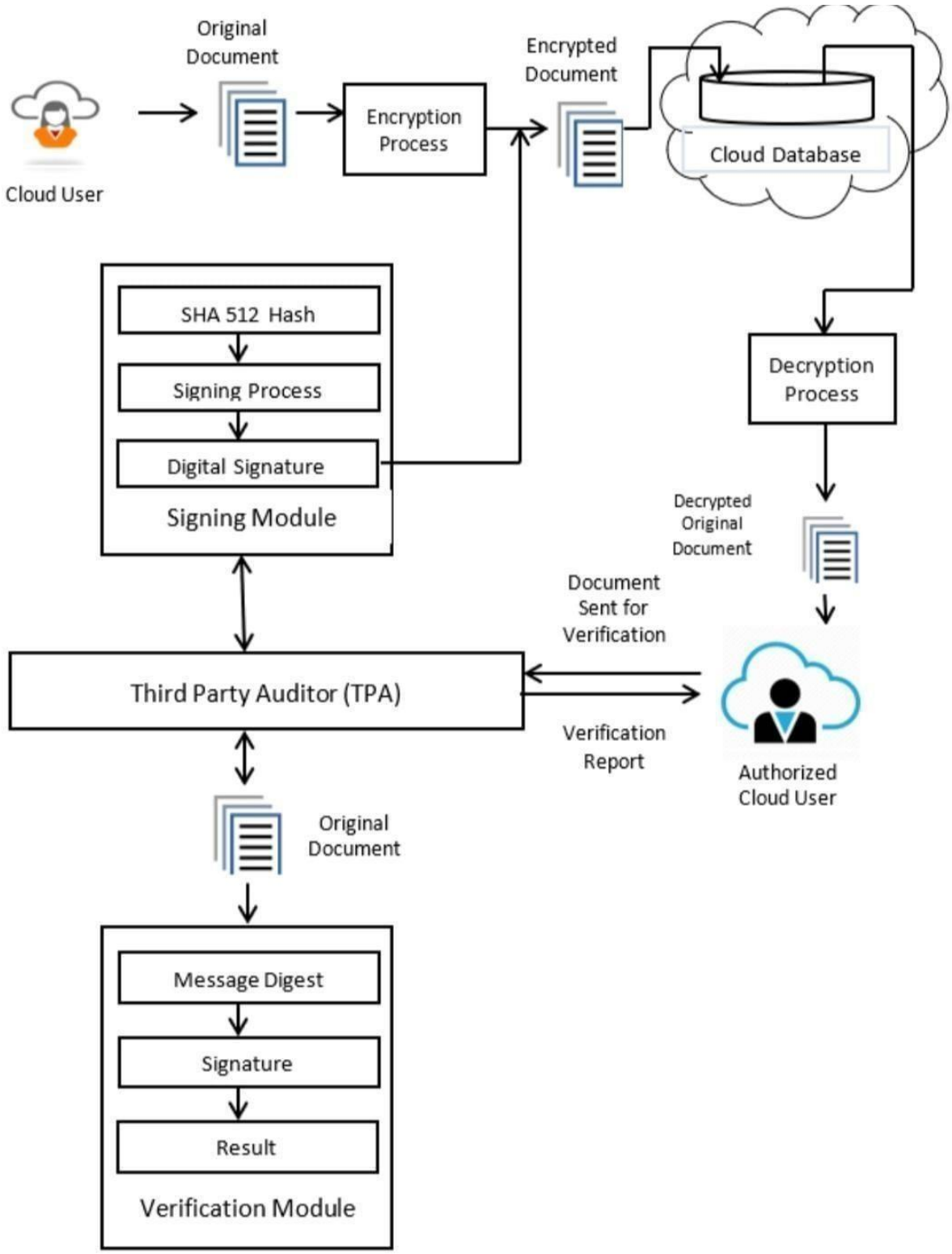
A digital signature is a cryptographic technique that addresses these challenges by providing a secure means of verifying the identity of the sender and the integrity of the message. It functions similarly to a handwritten signature but with much higher levels of security, leveraging public key cryptography. By using a pair of keys—a private key for signing and a public key for verification—digital signatures ensure that the information originates from a trusted source and has not been tampered with during transmission.

This paper explores the implementation of digital signatures in the context of secure transactions. It delves into the working mechanisms of commonly used algorithms such as RSA (Rivest-Shamir-Adleman) and ECDSA (Elliptic Curve Digital Signature Algorithm), explaining how they enable secure signing and verification processes. Moreover, it discusses the role of Public Key Infrastructure (PKI) in managing digital certificates and establishing trust between transacting parties.

The objective of this work is to demonstrate how digital signatures can be effectively applied to safeguard electronic transactions against unauthorized access and data manipulation. Through practical implementation and analysis, this study emphasizes the necessity of digital signatures in creating secure, transparent, and legally valid digital communications. Digital signatures work by using a private key to sign the message and a corresponding public key to verify the signature. If the content is altered in any way after being signed, the signature becomes invalid, thereby alerting the recipient to potential tampering. This makes digital signatures vital for maintaining data integrity, authentication, and non-repudiation—key pillars of information security.

3.

FLOW CHART


Original Document
Encryption Process
Encrypted Document
Cloud Database
Cloud User
SHA 512 Hash
Signing Process
Digital Signature
Signing Module
Decryption Process
Decrypted Original Document
Document Sent for Verification
Third Party Auditor (TPA)
Verification Report
Authorized Cloud User
Original Document
Message Digest
Signature
Result
Verification Module

4. SOFTWARE REQUIREMENTS

- **Operating System**: Windows 10/11, macOS, or any Linux distribution (Ubuntu recommended)
- **Programming Languages:** Python 3.8 or higher
- **Libraries/Frameworks**:cryptography or PyCryptodome – for RSA key generation, signing, and verification
- **Development Environment:** Visual Studio Code, PyCharm, or Jupyter Notebook
- **CLI Tools (optional):** OpenSSL – for manual certificate and key generation
- **Web Framework (optional):** Flask or Django – if building a web interface for transaction signing
- **Browser Extension:** Postman or REST Client – for testing API endpoints (if applicable)
- **Package Manager:** pip – for installing Python libraries
- **Version Control:** Git (with GitHub or GitLab for collaboration and backup)

5. HARDWARE REQUIREMENTS

- **Processor**: Minimum: Dual-core processor (e.g., Intel i3 or AMD equivalent)
- **RAM**: 4 GB (recommended 8 GB for better performance with larger files)
- **Storage**: Minimum 100–200 MB of free disk space.
- **Display:** Minimum: Standard 13" screen (recommended Full HD display for better readability and interface handling)
- **Internet Connection:** Required for downloading libraries, packages, and updates
- • **Input Devices:** Standard keyboard and mouse or touchpad
- **Optional Peripherals (for advanced security use cases):** Smart card reader for secure key storage and digital signing

6. METHODOLOGY

The methodology for implementing digital signatures in secure transactions involves several key stages, combining cryptographic operations and software development practices. The process can be broken down into the following steps:

1. Requirement Analysis:

- Identify the purpose of digital signatures within the context of secure transactions.
- Determine the scope of the system (e.g., document signing, message authentication, etc.).
- Choose appropriate algorithms such as **RSA**, **ECDSA**, and **SHA-256** for signature and hashing.

2. Key Generation:

- Generate a public/private key pair for each user involved in the transaction.
- The private key is securely stored by the user and used for signing.
- The public key is shared and used by recipients to verify signatures.

3.Message Hashing:

- Before signing, the original message or transaction data is passed through a cryptographic hash function (e.g., SHA-256) to produce a fixed-size hash.
- This hash represents the content uniquely and is used instead of the raw message.

4.Digital Signature Creation:

- The hashed message is encrypted using the sender's **private key**.
- This encrypted hash is the **digital signature** and is sent along with the original message.

5.Integration with Application Logic:

- Develop a user interface (optional) to allow message input, signature creation, and verification.
- Use Python or other suitable languages and libraries for implementation.
- Optional: Add a web interface using Flask or Django for demonstration.

7. IMPLEMENTATION

STEP 1: Key Pair Generation

Objective: Create a pair of asymmetric keys (private and public).

Theory:

- The **private key** is used for signing.
- The **public key** is shared for verifying the signature.

CODE:-

```
from cryptography.hazmat.primitives.asymmetric import rsa  private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048) public_key = private_key.public_key()
```

STEP 2: Create a Digital Signature

Objective: Sign the message using the private key.

Theory:

1. Hash the original message.
2. Encrypt the hash with the sender's private key → Signature.

CODE:-

```
from cryptography.hazmat.primitives import hashes, serialization from cryptography.hazmat.primitives.asymmetric import padding

message = b"Transaction: Transfer $5000"  signature = private_key.sign(  message,  padding.PSS( mgf=padding.MGF1(hashes.SHA256()),  salt_length=padding.PSS.MAX_LENGTH
  ),
  hashes.SHA256()
)
```

STEP 3: Verify the Signature

Objective: Check if the received message + signature are valid. **Theory:**

1. Decrypt the signature using sender's public key → original hash.
2. Hash the received message again.
3. Compare both hashes.

CODE:-

```
from cryptography.exceptions import InvalidSignature try:
public_key.verify( signature,
message, padding.PSS(
mgf=padding.MGF1(hashes.SHA256()),
salt_length=padding.PSS.MAX_LENGTH
),
hashes.SHA256()
)
print(" Signature is
VALID") except
InvalidSignature:
print(" Signature is INVALID")
```

OUTPUT Signature is VALID

STEP 4: Tamper Detection (Optional Test) Modify the message and try verification again. It should fail.

CODE:-

```
tampered_msg = b"Transaction:
Transfer $50,000" try:
public_key.verify( signature,
```

```
tampered_msg,        padding.PSS(
mgf=padding.MGF1(hashes.SHA256()),
salt_length=padding.PSS.MAX_LENGTH
      ),
      hashes.SHA256()
   )
   print("Signature valid (unexpected!)") except InvalidSignature:
print("Tampering detected!
Signature is INVALID")
```

OUTPUT: Tampering detected! Signature is INVALID **STEP 5: Visualize Signature Time & Size (RSA)**

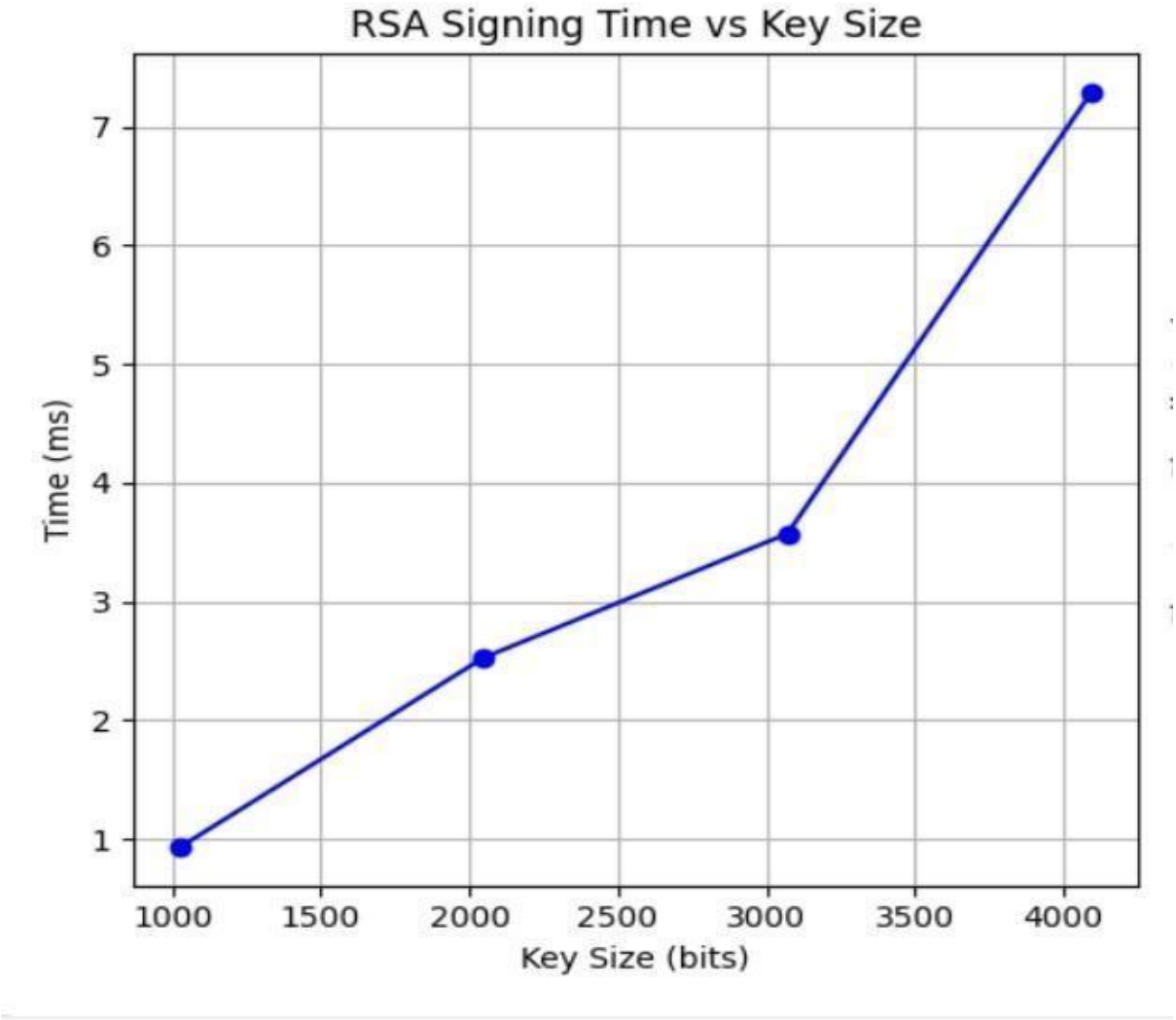
Python Code for Graphs:

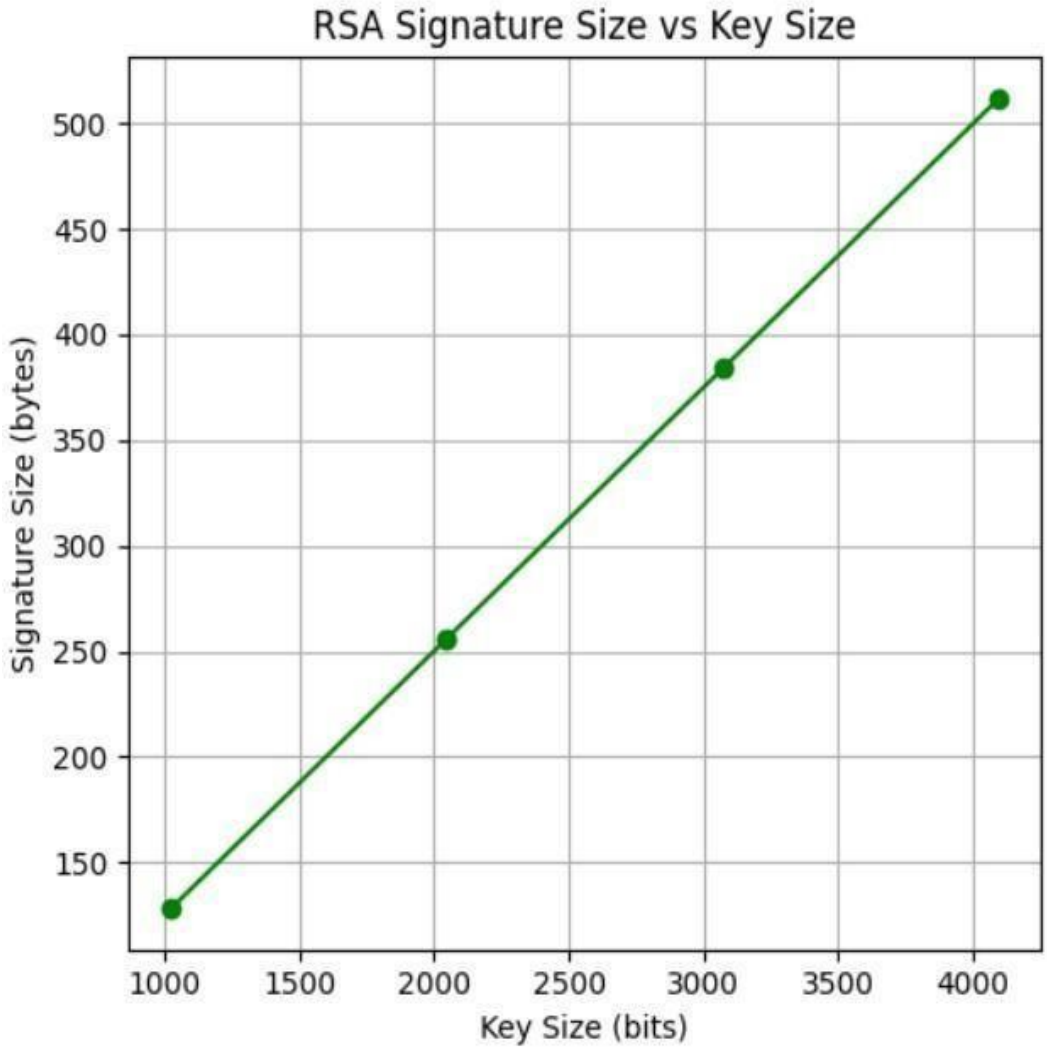
```
import matplotlib.pyplot as plt import  time
key_sizes = [1024, 2048, 3072, 4096] sign_times = []
sig_sizes = []  for size in key_sizes:
priv_key = rsa.generate_private_key(public_exponent=65537,  pub_key
= priv_key.public_key()

start = time.time()  sig = priv_key.sign( message,
padding.PSS(mgf=padding.MGF1(hashes.SHA256()), hashes.SHA256() )  end =
time.time() sign_times.append((end sig_sizes.append(len(sig))
# Plot
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1) - start)
* plt.plot(key_sizes, sign_times, marker='o', color='blue')
plt.title("RSA Signing Time vs Key Size")
plt.xlabel("Key Size (bits)")  plt.ylabel("Time (ms)")
plt.grid(True)  plt.subplot(1, 2, 2)  plt.plot(key_sizes,
sig_sizes, marker='o', color='green')  plt.title("RSA
Signature Size vs Key Size")  plt.xlabel("Key Size
```

(bits)") plt.ylabel("Signature Size (bytes)")

plt.grid(True) plt.tight_layout() plt.show(

8. RESULT




RSA Signature Size vs Key Size
Signature Size (bytes)
500
450
400
350
300
250
200
150
1000
1500
2000
2500
3000
3500
4000
Key Size (bits)

9. CONCLUSION

The successful implementation of digital signatures in this project has reinforced the importance of cryptographic methods in securing digital transactions. As digital communication continues to replace traditional paper-based systems, the need for secure, verifiable, and trustworthy interactions is more critical than ever. Digital signatures, built upon the foundation of asymmetric cryptography and hashing algorithms, offer a reliable solution to this need by ensuring data integrity, authenticity, and nonrepudiation.

Throughout the project, the full life cycle of a digital signature system was explored and implemented — from key generation to message signing and verification. This hands-on experience demonstrated not only the theoretical framework behind digital signatures but also the practical challenges, such as key management, certificate validation, and secure algorithm implementation.

Moreover, the project showcased the essential role of Public Key Infrastructure (PKI) in realworld applications. By simulating or using digital certificates, we observed how trust is established between users and systems, forming the backbone of secure communications over untrusted networks like the Internet.

The project also shed light on vulnerabilities and best practices. For example, the misuse or poor storage of private keys can completely undermine the security of the system. Hence, it became clear that while digital signatures are powerful tools, their effectiveness is heavily dependent on secure implementation and maintenance.

On a broader level, this project highlighted the relevance of digital signatures in various sectors such as egovernance, banking, e-commerce, legal documentation, and healthcare. As more organizations adopt digital processes, implementing secure and scalable signature mechanisms becomes not just beneficial but necessary.

Lastly, the project serves as a stepping stone for future innovations — such as blockchain integration, biometric authentication, cloud-based signing services, and quantum-resistant cryptography. The scope for enhancement and integration with other technologies is vast, ensuring that digital signatures will remain a cornerstone of cybersecurity in the digital age.