International Journal of Multidisciplinary Research and Growth Evaluation.

Developing a Practical Exercise on Digital Signature Encoding and Decoding

Do Manh Dung
Faculty of Fundamental Technical, AD-AF Academy of Vietnam, Son Tay, Ha Noi, Vietnam

* Corresponding Author: **Do Manh Dung**

Article Info

ISSN (online): 2582-7138
Volume: 06
Issue: 01
January-February 2025
Received: 15-12-2024
Accepted: 16-01-2025
Page No: 1957-1961

Abstract
Digital signatures are one of the most important security technologies, ensuring the authenticity, integrity, and non-repudiation of data in digital environments. This paper presents the development of a practical exercise on digital signature encoding and decoding, helping students understand the working principles and practical implementation of digital signatures. The exercise includes the following steps: generating public and private key pairs, digitally signing data using the private key, and verifying the signature using the public key. The exercise utilizes the Python programming language and popular cryptographic libraries to demonstrate the implementation process. The results of the exercise enable learners to master digital signature encoding and decoding techniques and apply them in information security systems.

Keywords: Digital signature, public key, private key, encryption, decryption, information security

Introduction

In the context of digital transformation and the rapid development of information systems, data security has become a critical factor in ensuring the safety, integrity, and authenticity of information. Digital signatures are one of the most important technologies in the field of information security, widely used to verify identities, protect data, and prevent forgery. According to Stallings' research, digital signatures are built on asymmetric encryption systems, where a private key is used to sign data and a public key is used to verify the signature [7]. This is an effective security method in electronic transactions, digital contracts, and online document authentication.

Digital signatures not only help protect information but also play a crucial role in building trust between parties involved in online transactions. According to research by Menezes, van Oorschot, and Vanstone [4], digital signature systems ensure three main security properties: Authentication: Ensuring that data comes from a legitimate source. Integrity: Ensuring that data is not altered during transmission. Non-repudiation: The signer cannot deny having created the signature. In e-commerce systems, e-government, and digital banking, digital signatures are used to verify the identities of participating parties, ensuring that documents and transactions are legitimate and not forged [3].

Digital signatures are built on public-key encryption algorithms, the most common of which are:

- RSA (Rivest-Shamir-Adleman): Developed in 1977, RSA is one of the first public-key encryption algorithms and is still widely used in digital signatures. RSA operates based on the difficulty of factoring large integers [6].
- DSA (Digital Signature Algorithm): A digital signature standard developed by the U.S. National Institute of Standards and Technology (NIST), using encryption based on the discrete logarithm problem [1].
- ECDSA (Elliptic Curve Digital Signature Algorithm): An improved version of DSA, using elliptic curves to enhance security and reduce key size [2].

Each algorithm has its own advantages and disadvantages, and the choice of algorithm depends on the specific requirements of the system.

Although digital signatures play an important role in information security, many students and IT engineers have not had the opportunity to directly practice these algorithms. Some challenges include: Lack of detailed practical materials on digital signature encryption and decryption processes. The complexity of algorithms and the requirement for foundational knowledge in cryptography. Limited availability of supporting tools in educational environments.

To address these challenges, this paper proposes a practical exercise to help students directly engage with digital signature algorithms through Python programming. Python is chosen because of its many encryption libraries, such as pycryptodome and cryptography, which simplify the implementation of digital signatures [2].

This paper aims to develop a practical exercise to help students understand the process of digital signature encryption and decryption, including: Instructions for generating public and private key pairs. Performing data signing with a private key. Verifying digital signatures with a public key. Applying digital signatures in data security. Through this exercise, students can improve their programming skills, gain a deeper understanding of how digital signatures work, and apply them in practice.

The remainder of the paper is organized as follows: Principles of digital signatures. Building the practical exercise. Results and evaluation. Conclusion.

Integrating digital signature encryption and decryption exercises into the curriculum will help students master theoretical knowledge and practical skills in information security. This paper provides a detailed guide to building an effective practical exercise, helping students gain a visual and practical understanding of digital signatures.

2. Principles of Digital Signatures

- Digital signatures are based on public-key cryptography, utilizing a pair of keys:

Private Key: Only the signer has the authority to use this key to create a digital signature.

Public Key: Shared widely to verify the signature.

- The process of creating and verifying a digital signature includes the following steps:

Creating a Digital Signature

Hash the Input Data: Use a cryptographic hash algorithm (e.g., SHA-256, SHA-3) to generate a hash value from the input data.

Encrypt the Hash Value: Encrypt the hash value using the private key to create the digital signature.

Send Data with Signature: Transmit the original data along with the digital signature to the recipient.

Verifying a Digital Signature

Decrypt the Signature: The recipient uses the sender's public key to decrypt the digital signature and retrieve the original hash value.

Rehash the Received Data: Hash the received data using the same hash algorithm.

Compare Hash Values: Compare the decrypted hash value with the newly computed hash value. If the two hash values match, the signature is valid, and the data has not been altered. If the hash values do not match, the data may have been tampered with or forged.

Digital signatures are a critical technology in information security, ensuring data integrity, authenticity, and non-repudiation. Algorithms such as RSA, DSA, and ECDSA have unique characteristics, making them suitable for specific applications. Understanding the principles and implementation of digital signatures is a fundamental foundation for building secure systems.

3. Building the Practical Exercise

3.1. Objectives

This practical exercise aims to help students master the concepts of digital signatures, understand how to generate keys, sign data, and verify digital signatures by practicing with the widely-used Python encryption tools. Upon completion, students will be able to:

Understand the working principles of digital signatures.

Know how to generate public-private key pairs.

Practice signing and verifying digital signatures on a file or message.

Apply digital signatures in real-world information security scenarios.

3.2. Tools and Software Used

Operating System: Windows

Python Software with Required Libraries: fitz, PIL, tkinter, PyPDF2, cryptography.

Original PDF File: The data file to be signed.

Signature Image File: signature.png.

3.3. Implementation Steps

Step 1: Generate a Secret Encryption Key

Generate a secret encryption key to be used for encrypting and decrypting data. This key is a 44-character byte string.

Step 2: Encrypt the Signature Image to Secure the Digital Signature

Encrypt the signature.png file and save it as signature_encrypted.bin to protect it from unauthorized access.

Step 3: Decrypt the Signature Image

Decrypt the signature_encrypted.bin file into signature_decrypted.png for use. Only individuals with the key can decrypt and use the digital signature.

Step 4: Embed the Digital Signature into the PDF

Read the signature from the decrypted file.

Determine the signature's position on the document.

Embed the signature into the last page of the PDF file and save it under a new name.

This structured approach ensures that students gain hands-on experience with digital signatures, from key generation to embedding signatures into documents, while understanding the underlying security principles.

3.4. Program Code

```
import fitz # PyMuPDF
from PIL import Image, ImageTk
import tkinter as tk
import io
from reportlab.pdfgen import canvas
from reportlab.lib.utils import ImageReader
import PyPDF2
from cryptography.fernet import Fernet

# Generate encryption key (run once and save)
def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
    key_file.write(key)
```

```
# Load key from file
def load_key():
    return open("secret.key", "rb").read()

# Encrypt the signature image
def encrypt_image(image_path, encrypted_path):
    key = load_key()
    cipher = Fernet(key)
    with open(image_path, "rb") as file:
    encrypted_data = cipher.encrypt(file.read())
    with open(encrypted_path, "wb") as file:
    file.write(encrypted_data)

# Decrypt the signature image
def decrypt_image(encrypted_path, output_path):
    key = load_key()
    cipher = Fernet(key)
    with open(encrypted_path, "rb") as file:
    decrypted_data = cipher.decrypt(file.read())
    with open(output_path, "wb") as file:
    file.write(decrypted_data)

# Determine the signature position on the PDF
def get_mouse_position(pdf_path):
    doc = fitz.open(pdf_path)
    last_page = doc[-1]
    pix = last_page.get_pixmap()
    img = Image.frombytes("RGB", [pix.width, pix.height],
    pix.samples)
    root = tk.Tk()
    root.title("Select Signature Position")
    img.thumbnail((600, 800))
    pdf_img = ImageTk.PhotoImage(img)
    canvas_widget = tk.Canvas(root, width=img.width,
    height=img.height)
    canvas_widget.pack()
    canvas_widget.create_image(0, 0, anchor=tk.NW,
    image=pdf_img)
    display_width, display_height = img.size
    real_width, real_height = pix.width, pix.height

def on_click(event):
    global x_real, y_real
    x_real = int(event.x * real_width / display_width)
    y_real = int(real_height - (event.y * real_height /
    display_height))
    root.destroy()

    canvas_widget.bind("<Button-1>", on_click)
    root.mainloop()
    return x_real, y_real, real_width, real_height

# Add signature to the last page of the PDF
def add_signature_to_pdf(input_pdf, output_pdf,
encrypted_signature):
    x, y, page_width, page_height =
    get_mouse_position(input_pdf)
    if x is None or y is None:
    print("Error selecting signature position.")
    return
    decrypted_signature = "decrypted_signature.png"
    decrypt_image(encrypted_signature,
    decrypted_signature)
    sig = Image.open(decrypted_signature)
    sig_width = 150
    sig_height = int((sig.height / sig.width) * sig_width)
    reader = PyPDF2.PdfReader(input_pdf)
    writer = PyPDF2.PdfWriter()
    for i, page in enumerate(reader.pages):
    packet = io.BytesIO()
    can = canvas.Canvas(packet, pagesize=(page_width,
    page_height))
    if i == len(reader.pages) - 1:

can.drawImage(ImageReader(decrypted_signature), x, y,
width=sig_width, height=sig_height)
    can.save()
    packet.seek(0)
    new_pdf = PyPDF2.PdfReader(packet)
    page.merge_page(new_pdf.pages[0])
    writer.add_page(page)
    with open(output_pdf, "wb") as output:
    writer.write(output)
    print(f"Signed PDF created at ({x}, {y}): {output_pdf}")

# Execute the process
generate_key() # Run once to generate the key
input_pdf = "filePDFgoc.pdf"
output_pdf = "filePDFky.pdf"
encrypted_signature = "signature_encrypted.bin"

# Encrypt the signature
encrypt_image("signature.png", encrypted_signature)

# Embed the signature into the PDF
add_signature_to_pdf(input_pdf,output_pdf,
encrypted_signature)
```

4. Results and Evaluation

4.1. Preparation

- Original PDF file to be signed

AD-AF ACADEMY OF VIETNAM

FACULTY OF FUNDAMENTAL TECHNICAL

LECTURE

Module: Principles of Communication

Lesson 16: Practice on Digital Signature Encoding and Decoding

Target Audience: Aviation Engineering Training

Academic Year: 2024-2025

January 14, 2025

HEAD OF DEPARTMENT

Dr. Le Ngoc Giang

HANOI, JANUARY 2025

Fig 1: Original PDF file to be signed

- Signature image file

Fig 2: Signature image file

4.2. Experimental Results

Step 1: Create a secret encryption key for use in data encryption and decryption.

This key only needs to be generated once and will then be saved to a file (secret.key) for future use.

The key must be saved to a file; otherwise, encrypted data cannot be decrypted later. Saving the key ensures that the data can be recovered in the future.

Note: Do not share the key with others if you do not want the data to be exposed. If the key is lost, the encrypted data cannot be recovered.

The key is a 44-character byte string:

Fig 3: The signature image is encrypted into a 44-character byte string

This key is used to:

Encrypt the signature image. When the digital signature needs to be protected, the signature image is encrypted before saving.

Decrypt the signature image when needed to retrieve the original image.

Step 2: Encrypt the signature image to secure the digital signature.

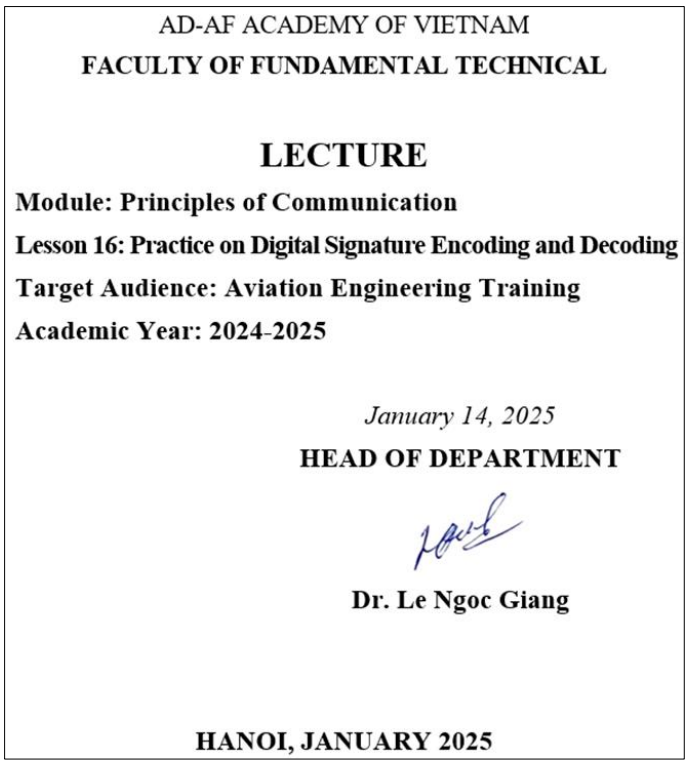
The file signature.png is encrypted and saved as signature_encrypted.bin.

Benefits:

Protects the signature image from unauthorized access.

No one can use the signature without the decryption key.

Fig 4: The file signature.png is encrypted and saved as signature_encrypted.bin


AD-AF ACADEMY OF VIETNAM

FACULTY OF FUNDAMENTAL TECHNICAL

LECTURE

Module: Principles of Communication

Lesson 16: Practice on Digital Signature Encoding and Decoding

Target Audience: Aviation Engineering Training

Academic Year: 2024-2025

January 14, 2025

HEAD OF DEPARTMENT

Dr. Le Ngoc Giang

HANOI, JANUARY 2025

Fig 5: The digital signature is embedded into the PDF document

Step 3: Decrypt the signature image.
The file signature_encrypted.bin is decrypted into signature_decrypted.png.
Benefits:
Only individuals with the key can use the digital signature.
Ensures security while allowing recovery when necessary.
Step 4: Automatically embed the digital signature into the PDF.
Read the signature from the decrypted file.
Create a temporary PDF file to contain the signature.
Embed the signature into the last page.
Save the modified page into a new PDF file.

Evaluation: After completing the steps in the practical exercise, the following results were achieved:
Successfully generated and stored the secret encryption key.
The signature image was encrypted and securely stored, and could only be decrypted using the corresponding key.
The digital signature was accurately embedded into the PDF document and can be easily verified.
The entire process was automated using Python code, saving time and ensuring accuracy.

4.2. Effectiveness Evaluation

The practical session is evaluated based on the following criteria:
Security: The use of encryption ensures that the digital signature cannot be forged or modified without the decryption key.
Accuracy: The digital signature verification process operates efficiently, ensuring reliable document authentication.
Automation: The entire process is programmed, minimizing manual errors and improving workflow efficiency.
Practical Applicability: The solution can be expanded for use in digital document authentication systems within enterprises and organizations.
Performance: The source code runs stably and processes medium-sized PDF files quickly.
In summary, this practical session provides a detailed and applicable process for securely and accurately signing digital documents.

5. Conclusion

The paper provides a detailed presentation of the process for developing the practical session on digital signature encryption and decryption, covering preparation, implementation, and result evaluation. By applying modern encryption algorithms such as RSA and ECC, the practical session helps learners understand the operating principles of digital signatures while enhancing their hands-on skills in data security and authentication.
Experimental results demonstrate that this approach ensures high reliability, maintaining the integrity and authenticity of digital documents. The practical session also enables learners to master digital signature processing techniques in real-world environments, particularly when working with electronic document systems.
In the future, this practical session can be expanded in the following directions:
Integration with advanced security protocols: Combining digital signatures with multi-factor authentication (MFA) to enhance security.
Application in real-world systems: Researching the implementation of digital signatures in smart contract systems and electronic transactions.
Algorithm optimization: Improving encryption algorithm performance for faster processing on resource-constrained devices.
Blockchain integration: Applying digital signatures in blockchain solutions to ensure data immutability and security.
This paper aims to contribute to improving training quality in the field of information security, providing students and researchers with a solid foundation to further develop practical applications in digital data security.

6. References

1. FIPS PUB 186-4. Digital Signature Standard (DSS). National Institute of Standards and Technology (NIST); c2013.
2. Hughes J, Bruce D. Practical Cryptography for Developers. Sebastopol, CA: O'Reilly Media; c2019.
3. Katz J, Lindell Y. Introduction to Modern Cryptography. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC; c2020.
4. Menezes AJ, van Oorschot PC, Vanstone SA. Handbook of Applied Cryptography. Boca Raton, FL: CRC Press; c2018.
5. Menezes AJ. Elliptic Curve Cryptography and Its Applications. Cham, Switzerland: Springer; c2020.
6. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1978;21(2):120-6.
7. Stallings W. Cryptography and Network Security: Principles and Practice. 7th ed. Boston, MA: Pearson; c2017.