Network Security (CSE350) Programming Assignment no. 4
(Project 4)

Programming language: Python
Platform: MS Windows
With Front-end implementation

Introduction

The project aims to develop a web server application that provides digitally signed degree certificates and grade cards upon request. These certificates are digitally signed by university authorities, ensuring their authenticity and integrity. The server handles requests containing the graduate's name and unique roll number, delivering the requested documents securely.

For running the code - run the following in the terminal - python3 appl1.py
Open the URL in any of the browser
There enter the following details of the graduate for requesting the graduate certificate and grade certificate

Implementations-

1. generate_watermark(roll, suffix)
Purpose: This function generates a PDF watermark containing information about the document issuance, such as the roll number and timestamp.
Parameters:
roll: The roll number of the student.
suffix: A suffix to differentiate between different types of watermarks (e.g., certificate watermark, grade card watermark).
Returns: The filename of the generated watermark PDF.

2. MERGEWWATR(doc_name, waterfile)
Purpose: This function merges a watermark PDF with the original document PDF.
Parameters:
doc_name: The filename of the original document PDF.
waterfile: The filename of the watermark PDF.
Returns: The filename of the merged PDF document.

3. GD(name, roll, suffix, pkey_registrar, pkey_director)
Purpose: This function generates a digitally signed document (certificate or grade card) with signatures from the registrar and director.
Parameters:

name: The name of the student.
roll: The roll number of the student.
suffix: A suffix to differentiate between different types of documents (e.g., certificate, grade card).
pkey_registrar: The private key of the registrar for digital signature.
pkey_director: The private key of the director for digital signature.
Returns: Tuple containing signatures from the registrar and director, document hash, and filename of the merged document.

4. verifysig(signature, message_hash, public_key)
Purpose: This function verifies the digital signature of a document.
Parameters:
signature: The digital signature to be verified.
message_hash: The hash of the document's content.
public_key: The public key corresponding to the private key used for signing.
Returns: True if the signature is valid, False otherwise.

5. index()
Purpose: This function renders the index HTML template.
Returns: The rendered HTML template for the index page which we have created and located in the templates section

6. get_graduate_info()
Purpose: This function handles the POST request containing graduate information, verifies the credentials, generates and verifies the digitally signed documents, and renders the download page.
Returns: The rendered HTML template for the download page or authentication failure message.

7. download_pdf(filename)
Purpose: This function handles the GET request to download a PDF file.
Parameters:
filename: The filename of the PDF file to be downloaded.
Returns: The PDF file as an attachment or an error message if the file is not found.

8. _main_
Purpose: This block of code runs the Flask application when the script is executed directly.
Returns: Starts the Flask application.
Each function serves a specific purpose in the web server application, contributing to the generation, verification, and delivery of digitally signed degree certificates and grade cards.

Some of the following methods which are used for -

1.Getting the proper GMT Date and Time: datetime.utcnow() is used in the code to acquire the proper GMT date and time. The current date and time in UTC is returned by this method. But it's crucial to make sure that the Flask application server's system time is precisely synced with a trustworthy time source, such an NTP server. Using the HTTPS protocol, which encrypts the data sent back and forth between the client and the server, can guarantee communication security.
{datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]} which is implemented in the code

2. Ensuring Only the Graduate Can Download: Additional information beyond the roll number, such as date of birth, can be used for authentication. Modify the get_graduate_info function to include this additional information in the authentication process. Or even hashed password can be used in which only original password is only known to the user
dob = request.form['dob'] used for asking the date of birth

3.Finding the Source of Shared Documents: Watermarks come in handy when trying to find the source of shared documents. To add watermarks to the PDFs that are created, alter the generate_document function. These watermarks may have distinct numbers or other information linked to the initial student.

waterfile = generate_watermark(roll, suffix)
merged_file = merge_with_watermark(doc_name, waterfile)

4.Access to Public Keys: Yes, access to public keys is necessary to verify the digital signatures applied by the university authorities. Each authority (Registrar and Director) should have their public key pair. These public keys can be stored securely and accessed during the signature verification process.
For sample this can be implemented for verifying
verified_registrar = verify_signature(sig_registrar, pdf_hash, pkey_registrar.public_key())
verified_director = verify_signature(sig_director, pdf_hash, pkey_director.public_key())

5. Bonus Points: Sequential Digital Signatures
Implementation: The document is digitally signed by two university authorities (Registrar and Director) sequentially. Each authority generates a digital signature for the document hash using their private key.
Security Consideration: Sequential digital signatures provide additional assurance of the document's authenticity and integrity. The signatures are verified using the respective public keys of the authorities.
generate_document function to include the signing process for both authorities.
# Signatures by Registrar and Director
    sig_registrar = pkey_registrar.sign(pdf_hash,
padding.PSS(padding.MGF1(hashes.SHA256()), padding.PSS.MAX_LENGTH),
hashes.SHA256())
    sig_director = pkey_director.sign(pdf_hash, padding.PSS(padding.MGF1(hashes.SHA256()),
padding.PSS.MAX_LENGTH), hashes.SHA256())

The developed web server application ensures the secure and reliable delivery of digitally signed degree certificates and grade cards to authorized users. By incorporating cryptographic techniques and secure communication protocols, the application maintains the integrity and authenticity of the issued documents, meeting the requirements for a trustworthy certification system.

OUTPUT-

Both the digital signatures of registrar and director are verified