CRYPTOGRAPHY LAB: IIIT - HYDERABAD

DIGITAL SIGNATURES:

This experiment was prepared by Srinath Nair (2018111001) and Vishnu Sai (2018101051) as part of their Assignment in the course Intro to Software Systems (ISS). The assignment was submitted on 16 April 2019 after working on it for over 3 weeks. (To see repository, click here)

WEEK-WISE SPLIT:

Week-1:

Assignment was presented and topics were displayed for selection. Form was filled and our team was assigned the topic Digital Signatures.

Week-2:

The theory behind Digital Signatures was explored thoroughly and the topic was understood by us so that we could go about and proceed with the implementation. This week also involved us understanding the previously made lab so that the implementation becomes easier.

Following this, a simple python script was implemented to show the working of digital signatures and verified with the TA.

Week-3:

Implementation of the lab was done. The framework used in this experiment is Flask. The backend was made and proper routing was done to finally get the web application implemented properly.

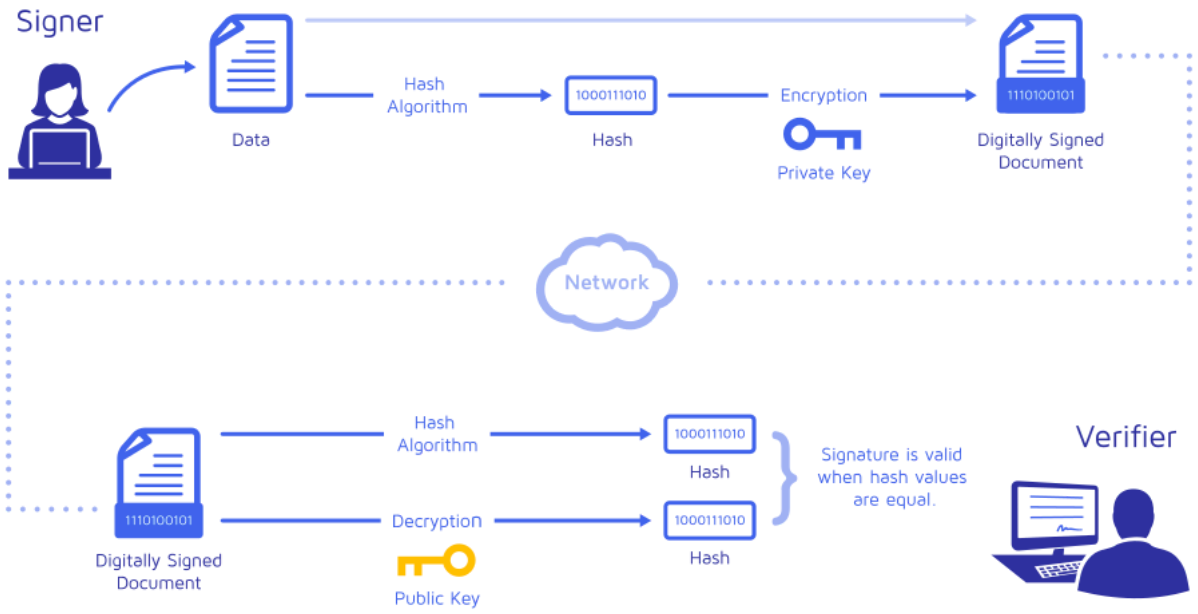
The final product was shown to the TA to get it evaluated.

THEORY:

A digital signature is a mathematical scheme for verifying the authenticity of digital messages or documents. A valid digital signature, where the prerequisites are satisfied, gives a recipient very strong reason to believe that the message was created by a known sender (authentication), and that the message was not altered in transit (integrity).

When a signer electronically signs a document, the signature is created using the signer's private key, which is always securely kept by the signer.The mathematical algorithm acts like a cipher, creating data matching the signed document, called a hash, and encrypting that data. The resulting encrypted data is the digital signature.

The buyer who receives the document also receives a copy of the sender's public key. If the public key can't decrypt the signature (via the cipher from which the keys were created), it means the signature isn't that of the sender, or has been changed since it was signed. The signature is then considered invalid.



IMPLEMENTATION:

The framework used in this web application is Flask. Required folder structure of static and templates have been created as required by Jinja to facilitate proper formatting of the web application we have created.

Implementation of databases has been done in the Quiz section of the experiment. The database implementation has been done using sqlite3. The user on clicking the submit button after choosing the appropriate option, would be redirected to a new page where the options he has selected would be displayed in the form of a table. The table is refreshed by removing it each time the quiz section opens up and by creating a new table each time the submit button is clicked.

The function gen_hash, hashes the message we enter in the text box and generates a SHA-1 encrypted digest in hexadecimal.

The function gen_keys, generates the public key and private key.

The next function is gen_DS, which is used to generate the digital signature for the message by encrypting the hash of the message with private key.

The function verify, takes the digital signature as input and decrypts it with the public key generated earlier by the function gen_keys and compares the decrypted message with the original hash which was generated by the gen_hash function. If the hashes are similar, then it means that the message was not tampered with and that the authenticity of the message is verified.

All the above function is under the class DSS which is imported into app.py from another python script dss.py.

The footer, sidebar and navbar have been separated are stored in an includes directory within the template directory where all the other html files are stored. The image files and javascript files are stored in the static directory.

How to run the experiment:

1. Enter the message in the Plaintext area.
2. Click SHA-1 to get the Hashed output.
3. Now generate the RSA Public key.
4. Now copy the Hashed output and give it as input to RSA.
5. Click Apply RSA to get the status.