**Encryption and Digital Signatures for Secure Communications**

**Abstract:**

Digital signatures are a cryptographic tool to sign messages and verify message signatures in order to provide proof of authenticity for digital messages or electronic documents [1]. Cryptography helps us to provide data confidentiality. Digital signatures are now widely used today in personal and industrial sectors.

**Introduction:**

Digital Signature Standard (DSS) was introduced by the National Security Agency (NSA) in 1994 [3]. DSS defines the Digital Signature Algorithm (DSA). Digital signatures are generated through DSA. In DSS there are 2 parties involved. One is senders and another is receivers. The sender generates the hash code from the original file, it encrypted with a private key and then sent over network to receiver side. Receiver then generates hash code from original file, decrypt with public key and validate the original data. If data is been modified by another party decrypted data will not match with actual one. This project will implement DSA for secure communication.

**Design and Implementation:**

The main program is written in a single class and coding language used is python—the main driver program created for both sender and receiver party. First, the program will compute a random 10-bit prime divisor and after that it will generate 20-bit random prime modulus such that primeModulus - 1) % primeDivisor not equal to 0. After that It will compute g = h\*\*(primeModulus - 1) / primeDivisor)) % primeModulus. Here h must be in between 1 and primeModulus-1. Then it will take input from the console for choosing a private key. The public key will also generate concurrently and this public needs to decrypt the signature. The program will then take input for which file it needs to be digitally signed. The hash generated by the SHA256 algorithm. After that the receiver party will ask for a file name and public key. When we enter the information, it will match the file hash and public key. For testing purpose only file with .txt extension have used.

The blue marked text is used for input. Red marked text used for the error. Before execution of this program we must create a sample.txt file for creating a digital signature. In this case, I have created a file name samplefile.txt and in this file can contain any information. There is another file I have created in the same directory named samplefile.MODIFIED.txt. This modified text will generate error as the original data have been altered.

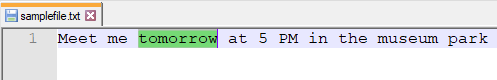


Figure 1: Sample of plaintext file

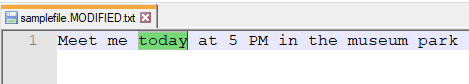


Figure 2: Sample of modified plaintext file

**Test and Results:**

During test I observed that encryption and decryption is working properly if we give correct information.

There are 3 possibilities that this program will detects the invalid signature of a file.

* Public key will not match
* Original data is not matching
* Public key and original data both are not matching

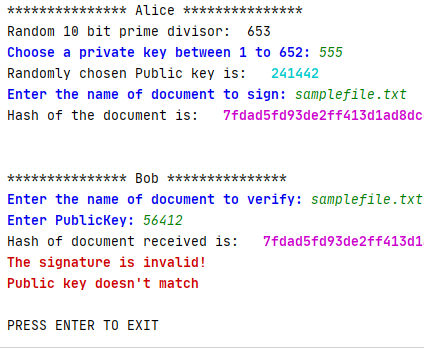


Figure 3: Invalid public key will give error

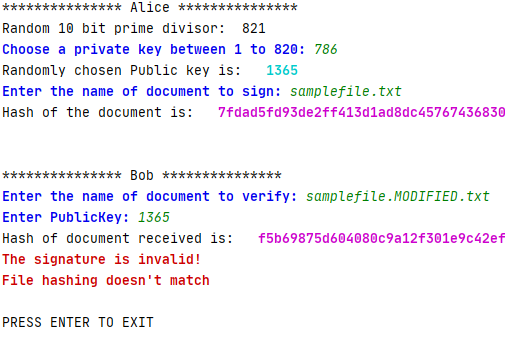


Figure 4: Modified or altered original data will give error

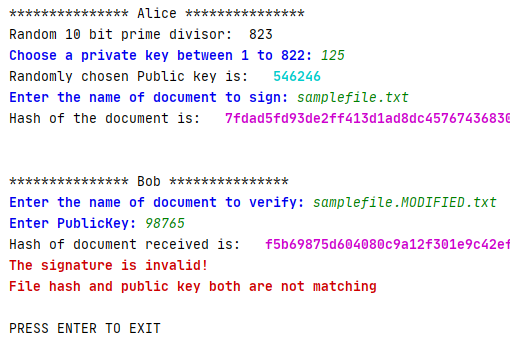


Figure 5: Invalid public key and altered data will give error

**Discussion:**

As per test analysis I can finalize that outcome of this program is good and I am very confident about my implementation.

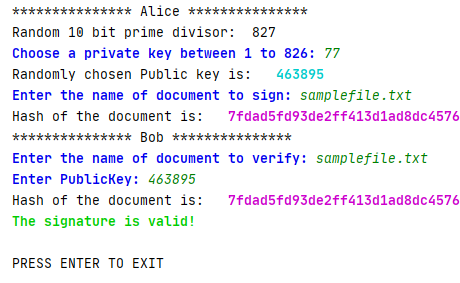


Figure 6: Output of the successful validation

**Conclusion:**

This assignment task implemented a DSA algorithm to sign and validate the data being transferred by both sender and receiver side. Future work would consider the implementation for RSA algorithm, and Elliptic Curve Digital Signature Algorithm.

**Reference:**

1. <https://wizardforcel.gitbooks.io/practical-cryptography-for-developers-book/content/digital-signatures.html>
2. <https://en.wikipedia.org/wiki/Digital_Signature_Standard>
3. <https://searchsecurity.techtarget.com/definition/Digital-Signature-Standard>
4. <https://www.includehelp.com/cryptography/digital-signature-algorithm-dsa.aspx>

**Part 2**

**Question 1:**

There are two kinds of URL we can notice during surfing web. One is http and another one is https. The little extra “s” is providing security other than http. When a web page is showing https in URL it means connection to that website is secure and encrypted. The technology is called SSL. SSL stand for Secure Sockets Layer. Generally, everyone can use the http without using the secure version (https) but when we fill up a from on some page or put some credit card details on the web unsecured information can be intercepted by a hacker. So, these types of problems can be overcome by SSL certificates. Website that's encrypted with SSL will form a connection with the web server, look at the SSL certificate, then bind the browser and the server for transfer the information securely and no one can see or capture what we are sending over the internet.

There are a number of different types of SSL certificates available to use - Domain Validated (DV) Certificates, Organization Validated (OV) Certificates, Extended Validation (EV) Certificates, Wildcard Certificates, Multi-Domain/ SAN/ UCC Certificates

|  |  |  |
| --- | --- | --- |
| SSL Certificate Name | Security Aspect | |
| Domain Validation (DV) Certificate | 2048-bit SSL Certificate signature  Encryption strength up to 256-bit | * Suitable for personal websites * Can be reissued as many times as needed during the validity period |
| Organization Validation (OV) | * Suitable for small/medium business * Unlimited number of server licenses |
| Extended Validation (EV) Certificate | * Suitable for large organization * Supports RSA and DSA encryption |
| Wildcard SSL Certificate | * Suitable for personal websites * Unlimited number of server licenses |
| Multi-Domain (SAN) SSL Certificate | * Suitable for personal and business * Can work with different types if domain |
| Multi-Domain Wildcard SSL Certificate | * Suitable for personal and business * Secure unlimited sub-domains with single certificate |
| UCC SSL Certificate | * Suitable for personal and business * Specially designed to work with Microsoft Exchange server |

Reference:

1. <https://aboutssl.org/type-of-ssl/>
2. <https://cheapsslsecurity.com/blog/types-of-ssl-certificates-explained/>
3. <https://www.cloudflare.com/learning/ssl/what-is-ssl/>
4. <https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/>
5. <https://blog.hubspot.com/marketing/what-is-ssl>

**Question 2:**

Using public key encryption HTTPS protects browser communication from being read or modified by third party channel. Browser prevents these types of manipulation by using digitally signed certificates. In cryptography, a certificate authority or certification authority (CA) is an entity that issues digital certificates [1]. CA verifies the identity of the owner from their secured database. SSL certificates are the digital file which follows RFC 5280 and which requires X.509 format to be conform. There are several numbers of CA available to validate security certificates.

SSL uses asymmetric private public key pair which is known as public key infrastructure (PKI). Public key distributed to the end users and CA uses the private key to check the authenticity of the certificate. There are some specific root (CA) such as Let's Encrypt who sign certificates but also sign for intermediary CA's. This follows what is called Chain of Trust, which is a chain of systems that trust each other [5]. When we browse any web page like google.com browser downloads all the details of the certificate. These details are what our browser uses to identify the validity. If any error occurs like expired certificate then the browser will show a warning message. Browser will verify it from the root CA if it is valid or not

Reference:

1. <https://en.wikipedia.org/wiki/Certificate_authority>
2. <https://www.venafi.com/blog/how-does-browser-trust-certificate>
3. <https://www.ssl.com/article/browsers-and-certificate-validation/>
4. <https://stackoverflow.com/questions/188266/how-are-ssl-certificates-verified>
5. <https://security.stackexchange.com/questions/56389/ssl-certificate-framework-101-how-does-the-browser-actually-verify-the-validity>