



**Somaiya Vidyavihar University K. J.  
Somaiya College of Engineering  
Department of Computer  
Engineering**

**Batch: A2 Roll No.:16010120030**

**Experiment No. 02**

**Title:** Application of RSA Algorithm for various security services like confidentiality, authentication, signature, non-repudiation and integrity.

**Objective:** To execute OpenSSL commands and to develop an application.

**Expected Outcome of Experiment:**

CO	Outcome
CO3	Apply Cryptographic Hash functions system security.
CO5	Illustrate Secure software design principles and apply them for secure software development.

**Books/ Journals/ Websites referred:**

<https://www.openssl.org/>

<https://github.com/openssl/openssl>

<https://en.wikipedia.org/wiki/OpenSSL>

Course: Information Security Lab Sem-VI Jan-May 2023 Page:



**Abstract:-**

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was first developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications. SSL is the predecessor to the modern TLS encryption used today.

A website that implements SSL/TLS has "HTTPS" in its URL instead of "HTTP." Originally, data on the Web was transmitted in plaintext that anyone could read if they intercepted the message. For example, if a consumer visited a shopping website, placed an order, and entered their credit card number on the website, that credit card number would travel across the Internet unconcealed.

SSL was created to correct this problem and protect user privacy. By encrypting any data that goes between a user and a web server, SSL ensures that anyone who intercepts the data can only see a scrambled mess of characters. The consumer's credit card number is now safe, only visible to the shopping website where they entered it.

SSL also stops certain kinds of cyberattacks: It authenticates web servers, which is important because attackers will often try to set up fake websites to trick users and steal data. It also prevents attackers from tampering with data in transit, like a tamper-proof seal on a medicine container.

**Related Theory: -**

OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. It is widely used by Internet servers, including the majority of HTTPS websites.

OpenSSL contains an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements basic cryptographic functions and provides various utility functions. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available.

The OpenSSL Software Foundation (OSF) represents the OpenSSL project in most legal capacities including contributor license agreements, managing donations, and so on. OpenSSL Software Services (OSS) also represents the OpenSSL project, for Support Contracts.

OpenSSL is available for most Unix-like operating systems (including Linux, macOS, and BSD) and Microsoft Windows.

**Implementation Details:****1. Enlist all the Steps followed and various options explored**

# open SSL

## Generating public / private key pair

### Generating Your Private Key

```
openssl genrsa -out aatmaj.key 2048
notepad aatmaj.key
openssl rsa -text -in aatmaj.key -noout
```

Private key will open like this and show the information about the key

aria-192-cfb8	aria-192-ctr	aria-192-ecb	aria-192-ofb
aria-256-cbc	aria-256-cfb	aria-256-cfb1	aria-256-cfb8
aria-256-ctr	aria-256-ecb	aria-256-ofb	base64
bf	bf-cbc	bf-cfb	bf-ecb
bf-ofb	camellia-128-cbc	camellia-128-ecb	camellia-192-cbc
camellia-192-ecb	camellia-256-cbc	camellia-256-ecb	cast
cast-cbc	cast5-cbc	cast5-cfb	cast5-ecb
cast5-ofb	des	des-cbc	des-cfb
des-ecb	des-ede	des-ede-cbc	des-ede-cfb
des-ede-ofb	des-ede3	des-ede3-cbc	des-ede3-cfb
des-ede3-ofb	des-ofb	des3	desx
idea	idea-cbc	idea-cfb	idea-ecb
idea-ofb	rc2	rc2-40-cbc	rc2-64-cbc
rc2-cbc	rc2-cfb	rc2-ecb	rc2-ofb
rc6	rc6-40	seed	seed-cbc
seed-cfb	seed-ecb	seed-ofb	serp-cbc
serp-cfb	serp-ctr	serp-ecb	serp-ofb

```
PS C:\Users\student1> openssl genrsa -out aatmaj.key 2048
PS C:\Users\student1> notepad aatmaj.key
PS C:\Users\student1> openssl rsa -text -in aatmaj.key -noout
Private-Key: (2048 bit, 2 primes)
modulus:
00:c2:e9:15:e3:d1:9f:7a:02:38:d1:61:90:21:bc:
8a:ef:8e:19:39:aa:f3:fa:1b:33:9e:88:8a:8a:8e:
cf:b8:c7:a6:1d:76:58:e9:dac9:fa:38:42:8d:6b:
51:51:04:c8:86:0a:7d:22:b6:b4:99:8c:8e:91:23:
d5:c6:51:1c:78:33:83:79:a8:49:e8:3f:33:87:87:
d6:7b:7b:3a:3b:28:88:15:55:c7:72:4c:62:93:e8:
private exponent:
1f:2b:bf:84:37:4d:2b:5c:a8:69:78:fd:de:4d:7b:
88:ea:de:3f:fb:30:af:bc:76:09:3c:29:9c:9f:97:
cc:39:04:df:ce:af:4b:73:e9:15:ed:a4:b5:56:2e:
```

### Generating Your Public Key

```
openssl rsa -in aatmaj.key -pubout -out aatmaj_public.key
notepad aatmaj_public.key
```

```
aa:de:cb:8c:a1:87:a5:3b:28
exponent:2:
00:01:ba:4a:5f:e9:1f:9c:3f:4c:92:cc:e3:56:26:
a9:a8:df:58:69:57:95:78:a8:c5:ad:c1:91:25:c3:
ff:68:ad:fe:08:d2:a8:fc:97:aa:3f:a6:1e:33:2d:
d3:d3:08:e2:34:40:06:80:83:43:56:58:28:2a:1e:
97:01:85:42:4f:92:47:39:31:99:5a:9c:58:f6:27:
6e:41:a9:4a:74:ed:c1:9f:e4:99:b6:12:87:88:83:
cc:27:83:97:dc:41:5f:6a:84:1c:93:1a:13:94:45:
d5:f9:0a:44:d1:4b:2b:db:d2:28:7d:ce:e8:d1:1c:
cd:4a:ff:c2:a7:b2:fb:eb:e9
coefficient:
23:e1:59:68:5b:9a:ef:65:13:a6:19:d1:5d:37:b1:
58:be:1c:84:35:99:9a:88:f2:69:db:f1:5a:ed:73:
d2:87:e7:1c:f7:e6:96:83:ff:31:3d:9e:7f:88:92:
55:db:d5:38:57:c6:d2:74:a2:88:fe:d3:bf:a1:63:
fe:53:32:82:6e:8c:8d:b6:f9:16:13:2d:dd:6d:89:
13:cc:67:a9:3f:86:3f:9c:2a:e9:d8:43:94:4c:55:
d8:3d:ff:18:a5:ab:2d:ff:5f:f1:6a:53:1b:d8:17:
8c:a9:6a:7f:8d:b5:8c:8e:98:f8:56:c2:8c:8d:6a:
81:35:de:3e:89:f8:1a:b6
PS C:\Users\student1> openssl rsa -in aatmaj.key -pubout -out aatmaj_public.key
writing RSA key
PS C:\Users\student1> notepad aatmaj_public.key
PS C:\Users\student1>
```

## Encrypt the file

Encrypting a text file using RSA using the above public key

```
openssl pkeyutl -encrypt -in plaintext.txt -inkey aatmaj_public.key -pubin -out encrypted.txt
notepad plaintext.txt
notepad encrypted.txt
```

```
P5 C:\Users\student1> openssl pkeyutl -encrypt -in plaintext.txt -inkey aatmaj_public.key -pubin -out encrypted.txt
P5 C:\Users\student1> notepad plaintext.txt
P5 C:\Users\student1> notepad encrypted.txt
P5 C:\Users\student1>
P5 C:\Users\student1>
P5 C:\Users\student1>
P5 C:\Users\student1>
P5 C:\Users\student1>
P5 C:\Users\student1>
```

The screenshot shows a Windows Notepad window titled "plaintext". The menu bar includes File, Edit, View, and icons for undo, redo, find, and settings. The text area contains several lines of Base64-encoded data:

```
-->)1B=QWU7SDAEIM_6w/L1LBI**CaAq0VVe"/X/BdDv/zBW~gEUdsATÜö"PmH#
06
S6_4tBSwBB"BMBE/|GXX^IëvRHsqOUKb
"BeIFBdB"B_u±qt">ÄT,X+äILByBBiq'-ãÁÉf/cphCTeß
EBRQQÖBBB?TB(rwa*)ÖBw-a4)IBES4{`BBIRBU);Gp7x7`SqB.CqÜM',"eFBB e3
wBBB4"u"-hyIAjXBdAmBgHrHUkGSUD-FB+^BBkte
```

Lets hash a file using SHA 256

```
openssl dgst -sha256 plaintest.txt
```

Generate a password secured certificate key. Then generate certificate using the key

```
openssl genrsa -des3 -out aatmaj_cert.key 2048
..Enter PEM pass phrase:
..Verifying - Enter PEM pass phrase:
notepad aatmaj_cert.key
openssl req -key aatmaj_cert.key -new -out domain.csr
openssl x509 -text -noout -in domain.crt
```

```
PS C:\Users\student1> openssl dgst -sha256 plaintext.txt
SHA2-256(plaintext.txt)= db35fb45d1ebc6015e9b10296a3ad6f942a7bcc83d752f70472b58d539937f83
PS C:\Users\student1> openssl genrsa -des3 -out aatmaj_cert.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
PS C:\Users\student1> notepad aatmaj_cert.key
PS C:\Users\student1> openssl req -key aatmaj_cert.key -new -out domain.csr
Enter pass phrase for aatmaj_cert.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Maharashtra

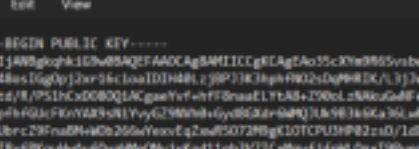
Locality Name (eg, city) []:mumbai
Organization Name (eg, company) [Internet Widgits Pty Ltd]:kjsce

Organizational Unit Name (eg, section) []:student

Common Name (e.g. server FQDN or YOUR name) []:hinnopettasus
```

```
PS C:\Users\student1> openssl x509 -signkey aatmaj_cert.key -in domain.csr -req -days 360 -out domain.crt
Enter pass phrase for aatmaj_cert.key:
Certificate request self-signature ok
, emailAddress=hippo@cemaiya.edu
```

### Generate a key for the certificate



```

-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQwFAAOCBgSNIIEGECAGAgZAs5c3KthR6SvsbW7OpLX
j4wv4BBS1Gg9pJ2or3F6Loa1D1H8Lz3BP3K3hgnF80Z5DgP8RDK/L3J3CpWaz
HnETrd/LP/PS1hCx0DBQZACgseYrhFF8naaELTY8a+Z06oLz5KcuGdFq/Z5ac
Y1dgpHrFQJkZFRVAYZ3vH1VyyGZ98WbhaGydb8G8a+9M9L3k5K36LwP8jv5V
a5FCUbcZ9Fru8Hw0b255aVveovEq2u05OTZ86gK0TCPUH82zsd/LdO3jwvJ
aekLT8FP0uHalaGyqy8Hk0BjYjKad11+u3V3J3CpWazHnETrd/LP/PS1hCx0DB
QZACgseYrhFF8naaELTY8a+Z06oLz5KcuGdFq/Z5acY1dgpHrFQJkZFRVAYZ3vH1
VyyGZ98WbhaGydb8G8a+9M9L3k5K36LwP8jv5Va5FCUbcZ9Fru8Hw0b255aVveov
Eq2u05OTZ86gK0TCPUH82zsd/LdO3jwvJaeaKLT8FP0uHalaGyqy8Hk0BjYjKad11+u3V3J3Cp
WazHnETrd/LP/PS1hCx0DBQZACgseYrhFF8naaELTY8a+Z06oLz5KcuGdFq/Z5ac
Y1dgpHrFQJkZFRVAYZ3vH1VyyGZ98WbhaGydb8G8a+9M9L3k5K36LwP8jv5V
a5FCUbcZ9Fru8Hw0b255aVveovEq2u05OTZ86gK0TCPUH82zsd/LdO3jwvJ
-----END PUBLIC KEY-----

```

Ln 1, Col 1    880 characters    100%    Windows (CR LF)    UTF-8

☐

```
PS C:\Users\student1> openssl dgst -sha256 -sign private.pem -out sign.sha256 plaintext.txt
Enter pass phrase for private.pem:
PS C:\Users\student1> openssl base64 -in sign.sha256 -out sign.bin
PS C:\Users\student1> openssl base64 -d -in sign.bin -out sign.sha256
PS C:\Users\student1> openssl dgst -sha256 -verify public.pem -signature sign.sha256 plaintext.txt
Verified OK
PS C:\Users\student1>
```



**Somaiya Vidyavihar University K. J.  
Somaiya College of Engineering  
Department of Computer  
Engineering**

**2. Explain the Importance of the approach followed by you**

**1. Public/Private Key Pair Generation:**

- a. Establishes the foundation for asymmetric cryptography.
- b. Enables secure communication by using a pair of keys - public and private.
- c. Public key can be freely distributed, while private key is kept secret.
- d. Allows encryption and decryption processes to occur using different keys, providing enhanced security.

**2. Public Key Encryption:**

- a. Facilitates secure data transmission over insecure channels.
- b. Utilizes the recipient's public key to encrypt the data.
- c. Only the recipient possessing the corresponding private key can decrypt and access the original data.
- d. Widely used in various applications like secure email communication, online transactions, and data protection.

**3. Hash Functions:**

- a. Converts variable-length input into a fixed-length hash value.
- b. Used for data integrity verification and digital signatures.
- c. Produces a unique hash value for unique input, minimizing the chance of collisions.
- d. Critical for password storage, ensuring that even if the hash is exposed, the original password is not easily revealed.

**4. Certificate Creation:**

- a. Certificates are a means of verifying the authenticity of entities in a digital environment.
- b. Certificates contain public keys and metadata, signed by a trusted Certificate Authority (CA).
- c. Enables secure communication by ensuring the legitimacy of public keys. d.

Essential for implementing secure web communication through protocols like HTTPS.

### **5. Digital Signatures:**

- a. Provides a mechanism for ensuring the authenticity and integrity of digital messages or documents.
- b. Created using the sender's private key and verified using the sender's public key.
- c. Guarantees that the message hasn't been tampered with and was indeed sent by the claimed sender.
- d. Crucial in securing transactions, software distribution, and other scenarios where data integrity and origin verification are paramount.

**Conclusion:-** Thus we have used OpenSSL to implement the RSA algorithm. We have done public and private key generation using openssl. Then we encrypted contents using public key encryption. We used SHA256 to hash a file. We also made digital signatures and made certificates.

Course: Information Security Lab Sem-VI Jan-May 2023 Page:  
**Somaiya Vidyavihar University K. J.**  
**Somaiya College of Engineering**  
**Department of Computer**  
**Engineering**

### **Post Lab Questions:**

2.1 In the RSA algorithm,  $p=7$ ,  $q=11$  and  $e=13$ , then what will be the value of  $d$ ?

**Step:****RSA Algorithm:**

- 1) Calculate value of  $n = p \times q$ , where  $p$  and  $q$  are prime no.'s
- 2) calculate  $\phi(n) = (p-1) \times (q-1)$
- 3) consider  $d$  as public key such that  $\phi(n)$  and  $d$  has no common factors.
- 4) consider  $e$  as private key such that  $(e \times d) \bmod \phi(n) = 1$
- 5) Cipher text  $c = \text{message i.e. } m^d \bmod n$
- 6) message = cipher text i.e.  $c^e \bmod n$

**Calculation**

$$p = 7, q = 11, e = 13$$

Use step 2 and 4 of RSA algorithm to calculate private key.

$$\phi(n) = (7-1) \times (11-1) = 6 \times 10 = 60$$

Now,

$$(e \times d) \bmod \phi(n) = 1$$

$$(13 \times d) \bmod 60 = 1$$

$$d = 37$$

So, key of A = 37

Source

<https://testbook.com/question-answer/in-rsa-algorithm-if-p-7-q-11-and-e-13-then--5ee785c420f2df0d16a93d2c>

## 2.2 Discuss various cryptanalysis attacks possible to be carried out on RSA

### 1. Brute Force Attack:

Description: Trying all possible private keys until the correct one is found.

Feasibility: Impractical due to the large key sizes used in RSA. The security of RSA relies on the difficulty of factoring large numbers, and the time required for a brute-force attack grows exponentially with key size.

### 2. Factorization Attacks:

Description: Attempts to factorize the public modulus ( $N$ ) to derive the private key.

Feasibility: The most significant threat to RSA. Efficient algorithms like the General Number Field Sieve (GNFS) and Elliptic Curve Factorization (ECM) are used for factorization. This is why choosing large prime numbers for RSA keys is crucial to resist factorization attacks.



### 3. Timing Attacks:

Description: Exploits variations in the time taken to perform cryptographic operations.

Feasibility: Can be successful if there are observable differences in the execution time of the RSA algorithm for different inputs. Implementing constant-time algorithms and protections against side-channel attacks can mitigate this risk. **4.**

### Common Modulus Attacks:

Description: Exploits situations where multiple users share the same modulus ( $N$ ) but have different public exponents.

Feasibility: Relevant in scenarios where different users generate their public keys using the same modulus. Careful key management practices are necessary to prevent common modulus attacks.

2.3 Comment on drawbacks of RSA. Discuss solution(s) over the same.

#### 1. Key Size and Performance:

- Drawback: As computing power increases, the recommended key sizes for RSA must also increase to maintain security. Larger key sizes result in longer computation times for encryption and decryption.
- Solution: Transition to more efficient public-key cryptosystems like Elliptic Curve Cryptography (ECC) that offer similar security with shorter key lengths. ECC provides a better performance-to-security ratio, making it a preferred choice for resource-constrained environments.

#### 2. Vulnerability to Quantum Computing:

- Drawback: Shor's algorithm, when executed on a large-scale quantum computer, can efficiently factorize large numbers, compromising the security of RSA. Quantum computers have the potential to break RSA by efficiently performing integer factorization.
- Solution: Transition to post-quantum cryptography algorithms, such as those based on lattice-based, code-based, hash-based, or multivariate polynomial cryptography. These algorithms are believed to be secure even in the presence of quantum computers.

#### 3. Padding Vulnerabilities:

- Drawback: The choice of padding schemes is critical in RSA, and vulnerabilities in padding can lead to security issues. For example, the PKCS#1 v1.5 padding scheme is vulnerable to certain attacks.
- Solution: Use more secure padding schemes, such as Optimal Asymmetric Encryption Padding (OAEP), which is designed to provide semantic security against chosen-ciphertext attacks. OAEP is more robust than older padding schemes and is recommended for modern RSA implementations.

