Tutorial #6
Security in Computing COSC2356/2357

(RSA Signature System)

Q1. Suppose Bob (the sender) wants to send a signed message **m=500** to Alice (the receiver). However, before sending the message he would like sign the message. When Alice receives the signed message, she would like to verify that the message is indeed from Bob. To facilitate signing and verification Bob generates public and private keys using **RSA encryption algorithm** and sends the public key to Alice. Bob uses parameter **p=113** and **q=89**, and chooses a suitable public key parameter **e=29**. How would Bob sign message **m=500**? How would Alice verify the signed message from Bob?

Answer:

Key Generation (by Sender)

- Bob Picks two prime numbers: $p = 113$ and $q = 89$
- Bob calculates: $n = p \times q = 113 \times 89 = 10057$
- Bob Calculates: $\varphi(n) = (p-1) \times (q-1) = 112 \times 88 = 9856$
- Bob chooses a prime number $e = 29$
- Bob creates a *Public-key*: $(n, e) = (10057, 29)$ and sends to Alice
- Bob Calculates d that is the multiplicative inverse of e modulo $\varphi(n)$ as follows:

$$d = e^{-1} \bmod \varphi(n)$$

$$d = 29^{-1} \bmod 9856$$

$$d = 7477$$

- Bob creates *Private-key*: $(n, d) = (10057, 7477)$

Signing (by sender)

- Bob signs the message ($m = 500$) (i.e. computes the signature) using private key $d = 7477$ as follows:
  $s = m^d \bmod n = 500^{7477} \bmod 10057 = 8065$
- *Bob sends (m, s) = (500, 8065) to Alice*

Verification (by receiver)

- Alice verifies using public key (n = 10057, e = 29) as follows:
  $m' = s^e \bmod n = 8065^{29} \bmod 10057 = 500$

As, $m = m'$ the message m is verified.

Q2. Suppose Bob (the sender) wants to send a signed message **m=1234** to Alice (the receiver). However, before sending the message he would like sign the message. When Alice receives the signed message, she would like to verify that the message is indeed from Bob. To facilitate signing and verification Bob generates public and private keys using **RSA encryption algorithm** and sends the public key to Alice. Bob uses parameter **p=131** and **q=97**, and chooses a suitable public key parameter **e=11**. How would Bob sign message **m=1234**? How would Alice verify the signed message from Bob?

Answer:

Key Generation (by Sender)

- Bob Picks two prime numbers: $p = 131$ and $q = 97$
- Bob calculates: $n = p \times q = 131 \times 97 = 12707$
- Bob Calculates: $\varphi(n) = (p-1) \times (q-1) = 130 \times 96 = 12480$
- Bob chooses a prime number $e = 11$
- Bob creates a *Public-key*: $(n, e) = (12707, 11)$ and sends to Alice
- Bob Calculates d that is the multiplicative inverse of e modulo $\varphi(n)$ as follows:

$$d = e^{-1} \bmod \varphi(n)$$

$$d = 11^{-1} \bmod 12480$$

$$d = 10211$$

- Bob creates *Private-key*: $(n, d) = (12707, 10211)$

Signing (by sender)

- Bob signs the message ($m = 1234$) (i.e. computes the signature) using private key $d = 10211$ as follows:
  $s = m^d \bmod n = 1234^{10211} \bmod 12707 = 6313$
- *Bob sends (m, s) = (1234, 6313) to Alice*

Verification (by receiver)

- Alice verifies using public key (n = 12707, e = 11) as follows:
  $m' = s^e \bmod n = 6313^{11} \bmod 12707 = 1234$

As, $m = m'$ the message m is verified.

(ElGamal Digital Signature System)

Q3. Suppose Bob (the sender) wants to send a signed message $\boldsymbol{m = 37}$ to Alice (the receiver). However, before sending the message he would like sign the message. When Alice receives the signed message, she would like to verify that the message is indeed from Bob. To facilitate signing and verification Bob generates public and private keys using **ElGamal encryption algorithm** and sends the public key to Alice. Bob chooses $\boldsymbol{p = 8081}$, $\boldsymbol{g = 2849}$, $\boldsymbol{x = 53}$. How would Bob sign message $\boldsymbol{m = 37}$? How would Alice verify the signed message from Bob?

Answer:

Key Generation (by sender)

- Bob selects a Prime $p = 8081$ and generator (e.g. primitive root) $g = 2849$ and chooses a private key parameter $x = 53$.
- Bob then generates a public key parameter y as follows:
  $y = g^x \bmod p = 2849^{53} \bmod 8081 = 6291$
- Bob creates *Public-key*: $(p, g, y) = (8081, 2849, 6291)$ and sends to Alice

Signing (by sender)

- Bob selects a random number k such that $1 \leq k \leq p - 2$, while satisfying $\boldsymbol{\gcd(k, p - 1) = 1}$. Let's pick $k = 11$. Obviously, $gcd(11, 8080) = 1$
- Bob computes signature parameters (r, s) as follows:

  $r = g^k \bmod p = 2849^{11} \bmod 8081 = 1158$

  $k^{-1} \bmod (p-1) = 11^{-1} \bmod 8080 = 6611$

  $s = k^{-1}(m - x.r) \bmod (p - 1)$
  $= 6611\ (37 - 53\ X\ 1158\ ) \bmod 8080$
  $= 6611\ (-61337) \bmod 8080$
  $= -405498907 \bmod 8080$
  $= -4107 \bmod 8080$
  $= 3973$
- Bob sends signed message $(m = 37, r = 1158, s = 3973)$

Verification (by receiver)

- Alice check if $1 \leq r \leq p - 1$. If true, then accepts the signature. Otherwise, rejects the signature.
- Alice computes verification parameters '$\boldsymbol{v}$' and '$\boldsymbol{w}$' as follows:

  $v = g^m \bmod p = 2849^{37} \bmod 8081 = 1874$

  $w = y^r . r^s \bmod p = 6291^{1158}.1158^{3973} \bmod 8081$
  $= ((6291^{1158} \bmod 8081).(1158^{3973} \bmod 8081)) \bmod 8081$
  $= 7107\ X\ 695 \bmod 8081$
  $= 4939365 \bmod 8081$
  $= 1874$
- Since $v = w = 1874$, signature is accepted.

Q4. Suppose Bob (the sender) wants to send a signed message $\boldsymbol{m = 23}$ to Alice (the receiver). However, before sending the message he would like sign the message. When Alice receives the signed message, she would like to verify that the message is indeed from Bob. To facilitate signing and verification Bob generates public and private keys using **ElGamal encryption algorithm** and sends the public key to Alice. Bob chooses $\boldsymbol{p = 83}$, $\boldsymbol{g = 79}$, $\boldsymbol{x = 29}$. How would Bob sign message $\boldsymbol{m = 23}$? How would Alice verify the signed message from Bob?

Answer:

Key Generation (by sender)

- Bob selects a Prime $p = 83$ and generator (e.g. primitive root) $g = 79$ and chooses a private key parameter $x = 29$.
- Bob then generates a public key parameter y as follows:

$y = g^x \bmod p = 79^{29} \bmod 83 = 15$

- Bob creates *Public-key*: $(p, g, y) = (83, 79, 15)$ and sends to Alice

Signing (by sender)

- Bob selects a random number k such that $1 \leq k \leq p - 2$, while satisfying $\boldsymbol{\gcd(k, p - 1) = 1}$. Let's pick $k = 11$. Obviously, $gcd(11, 82) = 1$
- Bob computes signature parameters (r, s) as follows:

$r = g^k \bmod p = 79^{11} \bmod 83 = 18$

$k^{-1} \bmod (p-1) = 11^{-1} \bmod 82 = 15$

$s = k^{-1}(m - x.r) \bmod (p - 1)$
$= 15\ (23 - 18\ X\ 29\ ) \bmod 82$
$= 15\ (-499) \bmod 82$
$= -7485 \bmod 82$
$= -23 \bmod 82$
$= 59$

- Bob sends signed message $(m = 23, r = 18, s = 59)$

Verification (by receiver)

- Alice check if $1 \leq r \leq p - 1$. If true, then accepts the signature. Otherwise, rejects the signature.
- Alice computes verification parameters '$\boldsymbol{v}$' and '$\boldsymbol{w}$' as follows:

$v = g^m \bmod p = 79^{23} \bmod 83 = 32$

$w = y^r . r^s \bmod p = 15^{18} . 18^{59} \bmod 83$
$= ((15^{18} \bmod 83).(18^{59} \bmod 83)) \bmod 83$
$= 23\ X\ 5 \bmod 83$
$= 115 \bmod 83$
$= 32$

- Since $v = w = 32$, signature is accepted.

TASK-1 (RSA Digital Signature using OpenSSL).

Say, you have a plain-text file, called "**plain-text.txt",** with your name and student ID in that file. Apply **openssl's RSA algorithm** to generate **2048** bit keys. Show that you can generate public and private keys and apply them to sign/verify. Make sure you apply a hash algorithm before signing the document.

Step-1: Run **openssl** command

```
openssl
```

Step-2: Generate a private key and store that private key on computer

```
genrsa -out private-key.pem 2048
```

Step-3: Generate the public key from the private key

```
rsa -in private-key.pem -pubout -out public-key.pem
```

Step-4: Hash the file "**plain-text.txt"** using **SHA256** and sign that file using the private key. The signed file **sign.sha256** is a binary file.

```
dgst -sha256 -sign private-key.pem –out sign.sha256 plain-text.txt
```

Step-5: Verify themessage:

```
dgst -sha256 -verify public-key.pem -signature sign.sha256 plain-text.txt
```

If the message is verified successfully, the following message will appear:

Verified OK