Week 4

Question 1: ELGamal

Bob (receiver) chooses

$p = 71, g = 69, x = 11$

Bob calculates:

$y = g^x \bmod p$

$= 69^{11} \bmod 71 = 11$

Bob sends to Alice the public key

$p = 71, g = 69, y = 11$

Alice (sender) has a message m to send

$m = 50$

Alice chooses a random number

$r = 23$

Alice calculates k

$k = y^r \bmod p = 11^{23} \bmod 71 = 28$

Alice calculates C_1 and C_2

$C_1 = g^r \bmod p = 69^{23} \bmod 71 = 42$

$C_2 = m * k \bmod p = 50 * 28 \bmod 71 = 51$

Bob receives C_1 and C_2

Bob calculates k

$k = C_1^x \bmod p$

$= 42^{11} \bmod 71 = 28$

Bob finds multiplicative inverse of k mod p

$k^{-1} \bmod p = 28^{-1} \bmod 71 = 33$

Bob decrypts C_2 to get the message m

$m = k^{-1} * C_2 \bmod p$

$= 33 * 51 \bmod 71 = 50$

Question 2: Paillier

Receiver chooses $p = 5$, $q = 7$

Receiver computes
$n = p * q = 5 * 7 = 35$
$n^2 = 1225$
$(p - 1) * (q - 1) = 4 * 6 = 24$

$\gcd(pq, (p - 1) * (q - 1))$
$= \gcd(35, 24) = 1$
=> hence p and q can be used for Paillier scheme (this is a condition shown in Lecture notes)

Receiver computes private key parameters
$\lambda = \text{lcm}(p - 1)(q - 1)$
$= \text{lcm}(4, 6) = 12$

Receiver computes private key parameter k:
$k = L(g^{\lambda} \bmod n^2)$ with $L(u) = (u - 1) / n$
$k = L(164^{12} \bmod 1225) = L(1121)$
$= (1121 - 1) / 35 = 32$

Receiver computes private key parameter μ:
$\mu = k^{-1} \bmod n$
$=$ multiplicative inverse of 32 mod 35
$= 23$

Receiver saves private key:
$(\lambda, \mu) = (12, 23)$

Receiver selects integer $g = 164$
(condition: order of g is a multiple of n, Lecture 4, slide 16)

Receiver sends the public key
$(n, g) = (35, 164)$

Sender selects a random number
$r = 17$

Sender encrypts plaintext $m = 1$
$C = g^m * r^n \bmod n^2$
$= 164^1 * 17^{35} \bmod 1225$
$= 127$

Receiver decrypts ciphertext c:
$m = L(c^{\lambda} \bmod n^2) * \mu \bmod n$
$= L(127^{12} \bmod 1225) * 23 \bmod 35$
$= L(1121) * 23 \bmod 35$
$= (1121 - 1) / 35 * 23 \bmod 35$
$= 32 * 23 \bmod 35$
$= 1$

OpenSSL Instructions

Run OpenSSL on Windows: https://youtu.be/Ts-gBfAW28c

On Mac, OpenSSL already installed, just open Terminal app, and type openssl, enter.

Using OpenSSL to generate RSA Private Key and Public Key

https://drive.google.com/file/d/1tBALzGyIT8THdvcFlMbNngWWr4A9z6No/view?usp=sharing

Using OpenSSL to encrypt a file

https://drive.google.com/file/d/1cO8oSQNCWwptT34SqYIwubr9yjRRJT7w/view?usp=sharing

Usin OpenSSL to decrypt a file

https://drive.google.com/file/d/1BnI1eRLi1qBuuFvvFm5-do9fnBrUu0Jf/view?usp=sharing

Link to slide

https://tinyurl.com/yb22rztz

Notice: in Task 2, Step 3 and Step 4, if Step 4 does not work for you, then go back to Step 3 and generate the pub-key.pem according to the following 2 commands

rsa -in key.pem -pubout -out pub-key.pem

rsa -in pub-key.pem -pubin -text -noout

Then, you can use the command in Step 4 as usual.