1 ElGamal Encryption

Recall ElGamal:

$KeyGen()$

- $p \in \mathbb{P}$
- $g \in \mathbb{Z}_p$ is a generator
- $x \xleftarrow{R} \{1, ..., p-1\}$
- $h = g^x$
- $pk = (p, g, h)$
- $sk = (x)$

$E(pk, m)$

- $y \xleftarrow{R} \{1, ..., p-1\}$
- Send the ciphertext $c = (g^y, m \cdot h^y)$

$D(sk, c)$

- Label the parts of the ciphertext as $c = (c_1, c_2)$
- Decrypt the message as: $c_2 \cdot ({c_1}^x)^{-1}$

1.1

Consider the following ElGamal setup:

- $pk = (G = \mathbb{Z}_{31}^*, g = 3, h = 6)$
- $sk = (x = 25)$

  Decrypt the following cipher texts:

- $(18, 19)$ **Proof.** [SOLUTION: 10] □
- $(22, 23)$ **Proof.** [SOLUTION: 14] □
- $(19, 2)$ **Proof.** [SOLUTION: 10] □

1.2

Ask a partner to select a message $m \in \mathbb{Z}_{31}^*$ and encrypt it for you. Decrypt it.

1.3

How many possible ciphertext pairs (c_1, c_2) are there for a given p and a specific value of x. How many valid values of x are there?

Proof. [SOLUTION:

- $(p-1)^2$
- $(p-1)$

] □

2 RSA without p or q

2.1

Suppose you have a private RSA key with p and q missing, but you have $\varphi(N)$. That is, you have:

$$\Big(N, e, \varphi(N)\Big)$$

Show that you can still find d and decrypt messages encrypted as $m^e \pmod{N}$.

Proof. [SOLUTION:

$$d = e^{-1} \bmod N$$

You can find e using the Extended Euclidean Algorithm.] □

2.2

Apply your solution to $\Big(N = 91,\ e = 5, varphi(N) = 72\Big)$

3 Reusing RSA Primes

In class, we showed that it's safe for everyone to use the same group and generator. Let's show that sharing the same modulus N doesn't work for RSA.

Suppose we try following:

- Generate $N = p \cdot q$ as in RSA.
- Generate *two* pairs of exponents (just do this step from normal RSA two separate times): (e_a, d_a) and (e_b, d_b)
- Publish $pk_a = (N, e_a)$ and give $sk_a = (N, e_a, d_a)$ to Alice.
- Publish $pk_b = (N, e_b)$ and give $sk_b = (N, e_b, d_b)$ to Bob.

Our goal is that anyone can encrypt to either Alice or Bob (using their corresponding public key), but neither Alice nor Bob can read messages intended for the other.

- Why can't we give p or q to Alice (or Bob)?
- Verify that Alice and Bob can still decrypt messages intended for them.

Now, show that the system is broken: Alice can actually decrypt any message for Bob.

(Hint: First, show that Alice can use her private key to find a number that is $\equiv 1 \mod \varphi(N)$. Then use problem 2.1 to show that she can find a decryption exponent d'_b that allows her to decrypt messages by Bob using normal RSA decryption. Is it necessarily true that $d_b = d'_b$?)

Proof. [SOLUTION: Use the Extended Euclidean Algorithm to find $e \cdot x + k \cdot \varphi(N) \cdot y = 1$.
] □

4 Even more Hardcore RSA

Show that in problem 3, Alice can actually factor N to recover p and q.

(This one is hard.)

Proof. [SOLUTION: See page 3 of https://crypto.stanford.edu/ dabo/papers/RSA-survey.pdf ("Twenty Years of Attacks on the RSA Cryptosystem")] □

5 RSA Blinding

5.1

Let's show that it's possible to sign a message using RSA without knowing what the message is.

Suppose Bob has a private key (N, e) and Alice wants him to sign m. She can do the following:

- Select a random $r \in \mathbb{Z}_N^*$.
- Calculate $r' = r^d$ This is the *blinding value*.
- Ask Bob to sign $m \cdot r'$ to get $s = Sign(sk, m \cdot r')$.
- Perform a multiplication on s to get a valid signature of $E(sk, m)$.

Find what they need to multiply in the last step to make this work, and argue why the whole procedure prevents you from knowing anything about m.

Why do you think this is called "blinding"?

Proof. [SOLUTION: s] □

5.2

Consider the following procedure.

- Select a random $r \in \mathbb{Z}_N^*$. This is the *blinding value*
- Perform a multiplication on m (probably using r) to get a value m'.
- Ask you to sign m'. to get $s = E(sk, m')$

Can you find a multiplication for the second step so that s is a valid signature for m without any additional calculations for them?

6 ElGamal Rerandomization

Recall that an ElGamal encryption is of the form

$$(g^y, m \cdot h^y)$$

where $x \xleftarrow{R}$ *Exponents* and $h = g^x$.

Take z to be any valid exponent. Show that it is possible for someone (who doesn't know x) to modify this so that the result (c_1, c_2) is the same as if the encryptor used $y' = y * z$ instead of y. That is, transform the original ciphertext into a valid ElGamal ciphertext of the format:

$$(g^{y'}, m \cdot h^{y'})$$

Verify that this decrypts.

7 ElGamal: Additively Homomorphic Encryption

Let $\mathcal{E}(m)$ be an algorithm that encrypts m to a specific person (i.e. choose values for G, g, and h and use the same ones every time). If you run it twice on the different messages m_1, m_2, you get:

- $a_1 = (g^{r_1}, m_1 \cdot h^{r_1})$
- $a_2 = (g^{r_2}, m_1 \cdot h^{r_2})$

Come up with an algorithm $\mathcal{E}'(a_1, a_2)$ that produces a valid encryption of the message $m_1 + m_2$

Note that if you were doing this in practice, you should make sure to rerandomize (see problem 6).

8 Programming

If you know how to program, implement one of these:

- Diffie-Hellman

- RSA
- ElGamal.

Talk to Conrad, Aaron, or Lucas if you'd like permission to work on this at the computers during study time.

(Do problems 1-8 first. We'll talk about the Chinese Remainder Theorem during study time if you finish all the other problems. :-P)

9 Chinese Remainder Theorem

Find numbers that satisfy the following congruences:

9.1

- $x \equiv 2 \pmod{5}$
- $x \equiv 1 \pmod{7}$

Proof. [SOLUTION: 22] □

9.2

- $x \equiv 9 \pmod{13}$
- $x \equiv 4 \pmod{24}$

Proof. [SOLUTION: 100] □

9.3

- $x \equiv 1 \pmod{2}$
- $x \equiv 1 \pmod{3}$
- $x \equiv 2 \pmod{5}$

Proof. [SOLUTION: 7] □

9.4

- $x \equiv 4 \pmod{7}$
- $x \equiv 3 \pmod{11}$
- $x \equiv 8 \pmod{13}$

Proof. [SOLUTION: 333] □

10 Chinese Remainder Theorem with Friends

Pick a partner you haven't worked with before.

Ask them to select a number x from 1 to 60 and tell you:

- $x \pmod 3$
- $x \pmod 4$
- $x \pmod 5$

Now, calculate x (if you get something outside of the range $1, ..., 60$, remember that you can add ay multiple of 60 and the congruences remain valid).

11 Speeding up RSA

Do an RSA encryption with a partner as in yesterday's homework. However, when you decrypt, calculate:

- $c^m \mod p$
- $c^m \mod q$

Now, use these values to calculate $c^m \mod N$ using the Chinese Remainder Theorem.