Assignment 2

This set consists of 4 questions worth a total of 60 marks

1 El Gamal Encryption (15 Marks)

Consider the following variant of El Gamal encryption. Let $p = 2q + 1$ with p, q prime. Let G be the group of squares modulo p (so G is a subgroup of $\mathbb{Z}_p^*$ of order q), and let g be a generator of G. The private key is (G, g, q, x) and the public key is (G, g, q, h), where

$$h = g^x \quad \text{and} \quad x \xleftarrow{\$} \mathbb{Z}_q.$$

To encrypt a message $m \in \mathbb{Z}_q$ choose a uniform $r \in \mathbb{Z}_q$, compute

$$c_1 := g^r \bmod p, \qquad c_2 := h^r + m \bmod p,$$

and output the ciphertext $\langle c_1, c_2 \rangle$.

Problem. Is this scheme CPA–secure? Prove your answer.

Answer.

A G group of squares modulo p means $\{x^2 \mod (p) : x \in Z_p\}$. So $g \in G$ is an element of $\mathbb{Z}_p^*$ of order q. This also means h, h^r, g, g^r are also in $\mathbb{Z}_p^*$ of order q.

So now for ciphertexts pairs c_1, c_2 we will try to show the probability that a bit/message is altered with non-negligble probability knowing that the h, g are squares modulo p.

Since p,q are coprime, then there are q in G values of that are squares modulo p and are generators of G. Rearrange $p = 2q + 1$ so that it is: $q = \frac{p-1}{2}$

Using c_2 we predict the next bit $b \in 0, 1$ by attacking by making a function $f(a)$ from Euler's criterion:

$$f(a) = a^q = a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \pmod p, & \text{if } a \text{ is a quadratic residue mod } p, \\ -1 \pmod p, & \text{if } a \text{ is a quadratic non-residue mod } p. \end{cases}$$

For some message $m \in Z_q$ with the next bit b. For $f(h^r + m) \implies f(h^r + b)$ Cases:

If b = 0, then $f(h^r + b) = f(h^r + 0) = f(h^r)$ and immediately the probability is that it is quadratic residue is $|G|/|Z*_p| = q/(p-1)$

If b = 1 then $f(h^r + b) = f(h^r + 1)$ and $h^r + 1$ cannot be a square and is not a quadratic residue of *mod p*. Therefore, the probability is that it is a quadratic residue is impossible, so Probability Quadratic residue mod p = 0. So we know it is not a quadratic residue

Thus, the probability of predicting the next bit

$$P(B = b) = \begin{cases} \text{if } b = 1, & \text{then } P(B = 1) = 1, \\ \text{if } b = 0, & \text{then } P(B = 0) = q/(p-1). \end{cases}$$

Therfeore, by the definition of IND-CPA for all efficient Adversary:

$$\text{Advantage} = |\Pr(B=1) - \Pr(B=0)| = |\frac{q-(p-1)}{(p-1)}|$$

So $(q-(p-1))/(p-1) >$ negligble and the attacker can determine the next bit with a high degree of certainty!!

Ergo, This scheme is not CPA-Secure.

2 RSA Encryption (14 Marks: 5 + 4 + 5)

Three users have RSA public keys $\langle N_1, 3\rangle$, $\langle N_2, 3\rangle$, and $\langle N_3, 3\rangle$ (so each uses $e = 3$) with $N_1 < N_2 < N_3$. To send the *same* message $m \in \{0,1\}^\ell$ to each party:

1. Choose uniform $r \xleftarrow{\$} \mathbb{Z}^*_{N_1}$ and compute

$$(r^3 \bmod N_1,\ r^3 \bmod N_2,\ r^3 \bmod N_3,\ H(r) \oplus m)$$

   where $H : \mathbb{Z}^*_{N_1} \to \{0,1\}^\ell$ and $\ell \gg n$ (the security parameter).

a Show that this scheme is *not* CPA–secure; an adversary can recover m from the ciphertext even when H is modeled as a random oracle.

b Propose a simple fix that yields CPA–security with ciphertext length $3\ell + O(n)$.

c Further improve your design so that the scheme remains CPA–secure but the ciphertext length is reduced to $\ell + O(n)$.

(Hint: the Chinese Remainder Theorem)

Answer.

Part a) The cipher text according to the algorithm is: $c = H(r) \oplus m$. Note, the Hash H modelled as a random oracle must be public to all users. According to the algorithm, all three users use the same remainder r to encrypt the plain text because it's given by the above algo

$$r^3 \bmod N_1 \implies r^3 \equiv a_1 \bmod N_1 \quad (1)$$
$$r^3 \bmod N_2 \implies r^3 \equiv a_2 \bmod N_2 \quad (2)$$
$$r^3 \bmod N_3 \implies r^3 \equiv a_3 \bmod N_3 \quad (3)$$

Where a_1, a_2, a_3 are some $\mathbb{Z}$. As a consequence of the same r^3, by the chinese remainder theorem, if N_1, N_2, N_3 are all pairwise coprime, this means when we find r by solving the above equations, we can decrypt the ciphertext by doing

$$H(r) \oplus c = H(r) \oplus (H(r) \oplus m) = m$$

.

Thus we have recovered m from the One Time Pad, and the algorithm is not CPA-secure!

Part b) Now we make all the remainders r in the algorithm different:

$$r_1^3 \equiv a_1 \bmod N_1 \tag{4}$$
$$r_2^3 \equiv a_2 \bmod N_2 \tag{5}$$
$$r_3^3 \equiv a_3 \bmod N_3 \tag{6}$$

Then a simple fix with ciphertext length $3l + O(n)$ complexity is broadcasting each user:

$$(H(r_1) \oplus m, H(r_2) \oplus m, H(r_2) \oplus m) = (c_1, c_2, c_3)$$

because they will know their respective secret r_n to decrypt some $c_n = H(r_n) \oplus m$.

This is CPA-secure because the remainders r_n are not dependent on each other like in part a)

Part c) To improve the ciphertext complexity we need to broadcast the shared secret s to all the users in the system.

1. We first choose uniform $r \xleftarrow{R} \mathbb{Z}_{N_1}^*$. We also make sure that user's do not use the the same remainder r.
2. Encrypt With each ith user's public key PK_i using the given $(N_i, 3)$ the shared secret s with the public key meaning: $\mathrm{PK}_i(s)$. This can only be decrypted with the ith user's RSA secret key. This takes $O(n)$ time
3. We send $\mathrm{PK}_i(s)$ to the ith user, as they can only decrypt it with the ith user's RSA secret key SK_i. So all three users now have the shared secret key s.
4. To broadcast an encrypted message now, we use $c = H(s) \oplus m$ and we send c to all three users. The cipher text is now l length.

Thus the total complexity is $l + O(n)$ time. Since we have a secret session key in the One Time Pad, that does not have any relationship to the other public values this is CPA-secure.

3 An Insecure Signature with Message Recovery (15 Marks: 7 + 8)

Let $T = (\mathcal{G}, F, I)$ be a one–way trapdoor permutation defined over $X := \{0,1\}^n$. Let H be a hash function from domain $\mathcal{M}_0$ to codomain X. Consider the signature scheme $\mathcal{S} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ defined on $(\mathcal{M}_0 \times X, X)$ by

$$\mathcal{S}(\mathsf{sk}; (m_0, m_1)) := \sigma \leftarrow I(\mathsf{sk}, H(m_0) \oplus m_1), \qquad \text{return } \sigma,$$
$$\mathcal{V}(\mathsf{pk}, (m_0, m_1), \sigma) := y \leftarrow F(\mathsf{pk}, \sigma), \qquad \text{accept iff } y = H(m_0) \oplus m_1.$$

1. Show that given (m_0, σ), where σ is a valid signature on (m_0, m_1), one can recover m_1.
2. Prove that the scheme is insecure, even when T is one–way and H is modeled as a random oracle.

Answer

<u>Part a</u>

Things public to us: Verifier (V), public key (PK), Hash function (H) and F and these ingredients are required by digital signatures.

Therefore, Given (σ, m_0) is a valid signatue, calculate $H(m_0)$

Then $y = F(\text{PK}, \sigma)$ this means for another message m_1

$y = H(m_0) \oplus m_1$

Therefore, we need to cancel out $H(m_0)$.

So $m_1 = y \oplus H(m_0) = (H(m_0) \oplus m_1) \oplus H(m_0)$

And we have recovered m_1.

QED

<u>Part b</u>

Things public to us: Verifier (V), public key (PK), Hash function (H) and F

Things Given to us:

TODO: We need to produce a valid (message(m_i), signature(σ_i)) pair, without knowing the secret key (SK).

Again calculate $H(m_0)$. Let's generate random signature $\sigma = 0, 1^n$. This will be our fake signature σ_i

Nice, let's get the corresponding y value. We do $y = F(\text{PK}, \sigma_i)$

Let's now generate the ith message: $m_i = y \oplus H(m_0)$

So now we have a fake (m_i, σ_i) pair.

So when we send (m_i, σ_i) to their other receiver, they will perform

$F(pk, \sigma_i) = y$

Then, when they match y it will be the same as $m_i \oplus H(m_0)$ and the receiver will find the signature to be valid!

Therefore the scheme is insecure even when T is one way and H is a random oracle.

4 DSA (15 Marks: 3 + 3 + 3 + 2 + 4)

To create parameters for the Digital Signature Algorithm (DSA) we first find primes p and q with $q \mid (p-1)$. Next we must find $g \in \mathbb{Z}_p^*$ of order q. Consider the algorithms below.

Algorithm 1

1. **repeat**
   a. choose $g \leftarrow \mathbb{Z}_p^*$;
   b. $h \leftarrow g^q \bmod p$;
2. **until** $(h = 1 \ \wedge \ g \neq 1)$;
3. **return** g.

Algorithm 2

1. **repeat**
   a. choose $h \leftarrow \mathbb{Z}_p^*$;
   b. $g \leftarrow h^{(p-1)/q} \bmod p$;
2. **until** $(g \neq 1)$;
3. **return** g.

Answer the following questions.

1. What happens in Algorithm 1 if g is chosen such that $\mathrm{ord}(g) = q$? Explain.
2. What happens in Algorithm 2 if h is chosen such that $\mathrm{ord}(g) = q$? (Recall that $g = h^{(p-1)/q} \bmod p$.)
3. Suppose $p = 64891$ and $q = 103$. How many loop iterations do you expect Algorithm 1 to execute before it finds a generator?
4. If p is 512 bits and q

Answer

Suppose:
$p = 7$
$q = 2$

Therefore for example
$\mathbb{Z}^* = 7 = \{1, 2, 3, 4, 5, 6\}$

a)
We go through each element $x \in \mathbb{Z}^*$, So x^2 mod 7.
Algo 1:
2^2 mod $7 = 4$
3^2 mod $7 = 2$
4^2 mod $7 = 2$
5^2 mod $7 = 3$
6^2 mod $7 = 1$

The algorithm goes through the elements in the $\mathbb{Z}_p^*$ multiplicative group of integers modulo p randomly.

Then it raises that particular elemnent $g \in \mathbb{Z}_p^* : g^q$.

and calculates $h = g^q$ mod (p).

It stops when $h == 1$ and g is not 1. Meaning not the trivial $1 = 1^q$ mod (p).

b)
$(p-1)/q = 3$
So we go through elements $x \in \mathbb{Z}^*$, So x^2 mod 7.
Algo 2:
2^3 mod $7 = 1$
3^3 mod $7 = 6$
4^3 mod $7 = 1$
5^3 mod $7 = 6$
6^3 mod $7 = 6$

The algorithm goes through the elements h in the $\mathbb{Z}_p^*$. Which is the multiplicative group of integers modulo p randomly.

Then it raises that particular elemnent $h \in \mathbb{Z}_p^* : h^{(p-1)/q}$.

and calculates $g = h^{(p-1)/q}$ mod (p).

It stops when $g == 1$.

c)
For $p = 64891, q = 103$ I expect algorithm 1 to run at the worst case scenario. This means for $p - 1$ elements in $\mathbb{Z}_p^*$, we will hit.

The generators (g) of order q mod (p) means g^q mod $(p) = 1$.

So every solution g, to the equation g^q mod $(p) = 1$ must have order $= q$.

This is because $q|(p-1)$ and both q and p are prime, then the number of generators of g that exist must be $q - 1$ to satisfy g^q mod $(p) = 1$. (Excluding the trivial element $g = 1$)

Small example:
if $p = 23$ and $q = 11$.

Then $G_{11} = \{1, 2, 4, 6, 8, 12, 13, 18, 22, 3, 11\}, |G_{11}| = 11$, and exlucding the trivial element 1, there are $|G_{11}| - 1 = 10$ $(g) \in G_{11}$ elements that satisfy the equation
$g^q \bmod (p) = 1$
$g^{11} \bmod (23) = 1$

So checking $2^{11} \bmod (23) = 1$. Indeed is true!.

Therefore, we would expect the probability to hit one of these generators to be $(q - 1)/(p - 1)$. REMEMBER TO EXLCUDE 1.

Thefore on average, the number of times we expect g to go through in algorithm 1 before hitting an number with order q is $(p - 1)/(q - 1) = 636.176$.

So we expect algorithm 1 to run approx. 636 times!

d)

If $p = 512$ bits and $q = 128$bit number then algorithm 1 will take $(2^{512} - 1)/(2^{128} - 1) \approx 2^{512}/2^{128} \approx 2^{384}$ times.

And the probability is $1/2^{284}$

Therefore it is not good to follow algorithm 1 in this case

For algorithm 2, we need to find the expected number of loops required to find $g = 1$ for algorithm 2. Since $q|(p - 1)$ then the size of the generator of subgroup of order q (G_q) is actually q.

Therefore the probability that we have to loop again is $q/(p - 1)$

So probability that we find $g = 1$, is $1 - q/(p - 1) = (p - 1 - q)/(p - 1)$ Therefore the probability to find a generator is very very close to 1!

e)

Therefore using the given hint for algorithm 2: $p = 64891$ and $q = 103$ The probability that algorithm 2 computes a generator $(g = 1)$ in it's very first loop is

$(p - 1 - q)/(p - 1) = (64891 - 1 - 103)/(64891 - 1) \approx 0.9984$!