CS346 - Asg 5 - David Ko - dpk326

May 1, 2015

Problem 1

(a) Given **ct**, we must create a different ciphertext **ct'** that, when decrypted, must be the same message *M*.

To get **ct**, we let $c_1 := g^y$, $c_2 := h^y m$. The *pk* is generated by *(p,g,h)* and *q* by *(p-1)/2*. And we know that Dec(*sk*, *ct*) = *M*.

So, lets say we uniformly choose at random another $x \in Z_q$ and generate a new g to get a new *(p,g)* generated by the Gen(1^n). With the new x, we get $h := g^x$ and a new *pk* := *(p,g,h)* and *sk* := *(p,g,x)*.

Now we uniformly choose at random $y \in Z_q$ to generate a new $c_1 := g^y$ and $c_2 := h^y m$ to return a new ciphertext **ct'**.

Now we have constructed a new and *different* ciphertext **ct'** such that **ct** $\neq$ **ct'**.

We have re-randomized the ciphertext using new exponents from Z_q and a new generator from G.

By re-randomizing the new ciphertext, we can efficiently and continuously construct a new ciphertext such that Dec(*sk*, *ct'*) = Dec(*sk*, *ct*) = *M*.

(b) We must construct an adversary that shows that El Gamal is not IND CCA secure.

1. We have an attacker and the challenger.

2. The challenger Gen(1^n) the public key and gives it to attacker as well as the decryption oracle. The challenger also chooses $x \in Z_q$ uniformly at random to generate $h := g^x$.

3. The attacker then queries 2 messages m_0 and m_1 and sends them to the challenger.

4. The challenger then choses a bit $b \in$ 0,1 uniformly at random and a $y \in Z_q$ and generates the ciphertext ct of m_b. The ciphertext is (c_1, c_2) where $c_1 := g^y$, $c_2 := h^y m$ (c_2 can also be rewritten as

$g^{xy}m$ since $h := g^x$) and then sends it to the attacker. With the ciphertext, the attacker can choose a random number z and modify the ciphertext to get *ct'* where $c_1 := g^{y+z}$, $c_2 := g^{(y+z)x}m$ since the attacker now knwos the values of g and g^y.

5. The attacker then can use the decryption oracle to decrypt this ciphertext *ct'* to get the original message m_b.

The attackers advantage in the IND-CCA game is 1 since the can get the value of a for the sk to decrypt to m_b.

Problem 2

1. There is an attacker and a challenger.

2. The challenger runs Gen(1^n) to generate *(p,g)* and q from (p-1)/2. The challenger chooses $x \in Z_q$ uniformly at random to get $h := g^x$. The challenger then sends the public key and decryption oracle to the attacker.

3. The attacker than sends 2 messages, m_0 and m_1 to the challenger.

4. The challenger then Enc(pk,M) by letting $M = (M_x, M_y) \in \mathbf{G}$ X $\mathbf{G}$. The challenger then chooses a random $r \in Z_q$. The ciphertext generated is $C_1 = g^r$ (like the unmodified El Gamal), $C_2 = A^r M_x$, and $C_3 = A^r M_y$. The message M is split into two messages M_x and M_y in this modified El Gamal.

5. This part is where the El Gamal fails the DDH assumption. C_2 and C_3 can be combined by the attacker to generate a C_x where $C_x = 2(A^r)$M. This is not IND CPA secure because the security of El Gamal depends on the security of G. G must be a prime order group and when you take the cross product **G** X **G** this fails, this breaks the security of the El Gamal encryption.

6. The attacker then recieves the ciphertext, and returns b which would indicate m_b.

The DDH assumption does not hold, therefore this modified EL Gamal is not secure.

The advantage of the attacker is Adv(A) = Pr[b=b'—d = 0] = $\frac{1}{2} + \varepsilon$.

Problem 3

In this problem, if the scheme is two time secure, then it is also one time secure. But if we construct a PPT A that breaks the two time secure scheme, then it also breaks the one time secure scheme.

1. The challenger does Setup(1^n) and sets the verification key to vk_1 and the signing key as sk_1. The challenger then sends the Signing oracle to the challenger.
2. The attacker than sends a message $m_0 \in 1, 0^n$ to the challenger.
3. The challenger signs the message using Sign(sk_1, m). Since this is the first message to be signed, the challenger then updates the signing key and verification key and chooses (sk_2, vk_2) and computes σ signature. After it signs the message m_0, it updates the **st** to (m, vk_2, sk_2, σ_1) and then sends the signature (m, vk_2, σ) to the attacker.
4. The attacker then creates a forgery and then sends it to the challenger to run Verify(vk_1, T, m). The challenger then outputs 0 if $m \neq m_i$. Else, 1.

This is where the one-time secure scheme ends since the attacker at max can make 1 query. This scheme is one-time secure, but lets look at if the attacker can make another query.

The attacker sends a second message m_1 to the challenger.

The challenger recognizes that this is not the first message sent, and updates the verification key and signing key to vk_3 and the sk_3.

What happens if m_1 is the same as m_0? Unlike Encrypt then MAC where if the same message is encrypted twice, the MAC part will be identical in both cases, since the state is updated the signing key and verficition key are not different from the prevous state, there fore avoiding the problem Encrypt Then MAC has.

The attacker now has to deal with a new signing key and verification key vk_3 and the sk_3 during this second query, so the result will be different from the previous query sincen the state is updated and keys are also updated.

From this we can see that this scheme is two-time secure.

Because the assumption that if this scheme is two-time secure, then it must also be one-time secure. If we can construct a PPT attacker that breaks the two-time secure scheme, then that PPT attacker can also break the one-time secure secure since a two-time secure scheme $\rightarrow$ the scheme is also one-time secure.

Problem 4
(a) Construct the encryption scheme ε.

Gen(1^n) runs Setup and generates the public key and the secret key *(pk, sk)*.
Enc(pk, m) takes the public key pk and a message $m \in M$ as input and outputs a ciphertext $ct \in C$ where $ct =$ (pk,m).
Dec(sk, ct) takes in the secret key sk and te cipher text and outputs a message $m \in C$.

Given that ε' is IND-CPA secure, we take a look at ε and see how secure it is.

We prove that ε' is IND-CPA secure by playng the security game.
1. The challenger generates a key k by running KeyGen(n) during Setup(1^n), where n is the security parameter.
2. We give the attacker Oracle access to the Enc(k).
3. The attack sends back 2 messages m_0 and m_1.
4. The challenger than choose a bit $b \in$ 0,1 uniformly at random and send the cipher text c from running Enc(k,m_b) back to the attacker.
5. The attacker is given Oracle Access to Enc(k,m) and the attacker outputs a guess *b'*.
The scheme ε' is secure if the Adv of the attacker is neglible. *negl(n)*.

So we see that ε' is IND-CPA secure from the security game under those conditions having a *negl(n)* advantage. Since it is IND-CPA secure, it is also IND-Random secure. So if ε' is IND-CPA secure, then we know that ε is IND-Random secure.
By contradiction, if there exists a PPT A that breaks ε', then

there exists a PPT B that breaks ε and ε is no longer IND-Random secure.

Showing that ε is not IND-CPA secure:

The attacker will win with an advantage of 1/2. Why is this?

1. The challenger runs Setup and generates a public key and secret key, **pk** and **sk** respectively. The challenger then sends the **pk** to the attacker.

2. The attacker then generates 2 message m_0 and m_1 uniformly at random from the message space and sends it to the challenger. The Challenger then choses a bit b at random from $0, 1$.

3. The challenger then runs Enc and generates a ciphertext ct from Enc(pk,m) and then sends it to the attacker.

4. The attacker did not himself generate the sk so he has no knowledge of what m_0 or m_1 is from just the ciphertext. The attacker just knows what the **pk** is and what the ct is. Because of this the attacker must just choose b' from either 0 or 1 because the attacker has no knowledge of what message was being used.

5. The attacker sends back *b'*. The challenger ouputs 0 if b $\neq$ b' or 1 if b = b'.

This is not IND-CPA secure because the attacker can beat the challenger 50% of the time. The attacker just has to choose either 0 or 1 and the attacker will get b right half the time resulting in an adv of:

Adv[A] = (Pr[b = 0 — b' = 0] + Pr[b = 1 — b' = 1]) - (Pr[b = 0 — b' = 1] + Pr[b = 1 — b' = 0]) = $\frac{1}{2}$.

(b) This scheme differs from the previous one in part (a) because in part (a) the message space was much smaller, allowing the message only to be either one or the other resulting in a advantage of 1/2.

But in this scheme, this message space is 0, 1, 2, ... n-1 which means that if the previous scheme has an adv of ϵ, then for each addition message available in the message space, the adv of the attacker would be reduce to $\frac{\epsilon}{n^2}$.

WHy is this? Lets take a look:

In the IND-CPA scheme before, the attacker would generate 2 messages m_0 and m_1 and send it to the challenger to encrypt. But since the message space is much larger in this currentn scheme, we would have to send the messages m_0, m_1, m_2... m_{n-1} thus reducing the adv of the attacker.

In the random game, the challenger generates 2 messages itself, so the attacker has no idea what they are; the attackerk only gets back the ciphertext, so all the attacker can do it guess either 0 or 1, but in this scheme, the attacker gets back too many messages and the adv of the attacker becomes:

Adv[A] = SUMMATION OF(Pr[b = 0 — b' = 0] + Pr[b = 1 — b' = 1]) + Pr[b = 3 — b' = 3])... Pr[b = n-1 — b' = n-1]) - SUMMATION OF (Pr[b = 0 — b' = 1] + Pr[b = 0 — b' = 2] + Pr[b = 0 — b' = 3].. and so on)) = which converges to $\frac{\epsilon}{n^2}$.