The University of Newcastle
School of Electrical Engineering and Computer Science

COMP3260 Data Security

GAME 9

8th May 2017

Number of Questions: 5
Time allowed: 50min
Total marks: 5

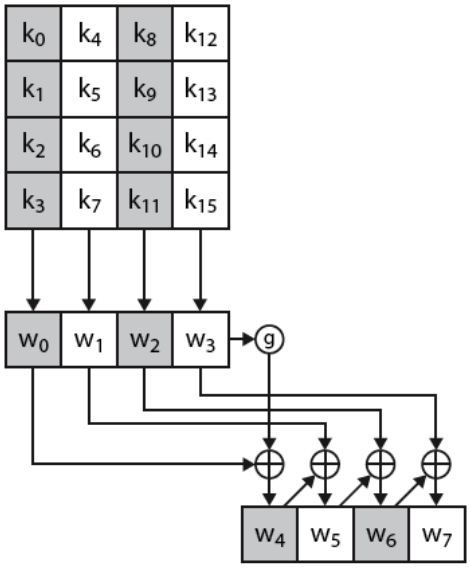
In order to score marks you need to show all working/reasoning and not just the end result.

	Student Number	*Student Name*
Student 1		
Student 2		
Student 3		
Student 4		
Student 5		
Student 6		
Student 7		

Question 1	Question 2	Question 3	Question 4	Question 5	Total

1. Describe, with the aid of a diagram how AES Key Expansion works. (Don't worry about the specific values of the S-Box or the specific values of the round constant)

Solution:



Take the first 4 bytes of the key to be the w0 of the expanded key, and the next 4 bytes to be the w1, the next 4 to be the w2 and the last 4 to be w3.
Run w3 through function "g" to get temp=g(w3). "g" works as follows:
1. shift the bytes to the left, such that the first byte is now the last byte.
2. run each byte through the AES S-Box.
3. XOR the result from step two with the Round constant
Now:
w4 = w0 ⊕ g(w3)
w5 = w1 ⊕ w4
w6 = w2 ⊕ w5
w7 = w3 ⊕ w3
w8 = w4 ⊕ g(w7)
etc.

2. For each of the following RSA operations, state if there is a known fast (polynomial or better) algorithm for solving it, and if so, name (or describe) the algorithm. (You have to get all of them right to get the mark)

a) Encryption/decryption

b) Calculating ϕ(n) from n.

c) Calculating *d* from e and ϕ(n)

Solution:

a) Yes,
   a. Fast Exponentiation $C = M^e \bmod n$

b) No
   a. you need to know p and q, and factoring n is not polynomial time

c) Yes
   a. Euclid's Extended Algorithm to compute multiplicative inverses
   b. $ed \bmod \phi(n) = 1$

3. Last week you managed to decrypt Ruby Cel's encrypted message by creating a lookup table of all the possible messages that could have been sent (the letters A-Z). Consider this slightly more realistic scenario: Suppose Bob wishes to establish a secure communications channel with Alice. They agree to use DES to encrypt their messages, and use RSA to send the session key. So, if Bob wanted to talk to Alice, he would choose a DES key, encrypt it with Alice's Public key and send it to Alice who decrypts it using her private key. They would then communicate using DES with the key that was sent to Alice.
Say Eve wanted to eavesdrop on the session. Knowing that Bob and Alice are using DES, she creates a lookup table of every possible DES key and the corresponding value when encrypted under Alice's public key (2^{56} pairs). She then is able to discover the session key by simply searching the lookup table, effectively reducing her effort of attacking the Public Key system to a brute force attack on a 56-bit key.
How could Bob and Alice modify their communications protocol to prevent Eve from carrying out this kind of probable message (table lookup) attack?
(There are several possible answers.)

Solution:

1. They could pad out the key with an agreed number of pseudorandom bits – then each key has a number of possible corresponding ciphertexts instead of just one, making the table lookup much more expensive
2. They could encrypt twice, first using their own private key, then using the other's public key
3. Some other creative solution

4. When using RSA, it is common to choose specific values of e for the sake of efficiency. Three popular choices of e are 65537 ($2^{16} + 2^0$), 3 ($2^1 + 2^0$), and 17 ($2^4 + 2^0$).
Consider this scenario: Alice is creating a chatroom to talk with her friends Bob, Charlie and Dan. All the messages in the chatroom are encrypted using AES. Alice sends the session key K to Bob, Charlie and Dan using RSA, encrypting K with their respective public keys, and they can all decrypt the message using their respective private keys.
Now it turns out that Bob, Charlie and Dan have all set their public key e, to the value 3 (but with different values of n). So Alice ends up sending $C_{Bob}=K^3 \bmod n_{Bob}$, $C_{Charlie}=K^3 \bmod n_{Charlie}$, and $C_{Dan}=K^3 \bmod n_{Dan}$.
Eve wants to eavesdrop on their chatroom, and so she intercepts C_{Bob} $C_{Charlie}$ and C_{Dan} in order to get the session key. Eve finds that n_{Bob} $n_{Charlie}$ and n_{Dan} are relatively prime, and thus she is able to use the Chinese Remainder Theorem to compute $X=K^3 \bmod (n_{Bob}n_{Charlie}n_{Dan})$.
Now, by the RSA specs, the message must be smaller than the n used, so we have $K < n$, for Bob's n, Charlie's n and Dan's n. This means that $K^3 < n_{Bob}n_{Charlie}n_{Dan}$, and thus Eve knows that $K = \sqrt[3]{X}$.
How can Alice send the message K to her friends without being vulnerable to the above attack? (I.e. how can she avoid encoding and sending the exact same message to each of her friends, but still have them all know what K is, so that she can talk to them)

Solution:
If Eve appends an agreed number of pseudorandom bits to K to generate a K' for each of her friends, then she can avoid sending the same message to each of her friends, and Eve is no longer able to use the Chinese Remainder Theorem to calculate K. Her friends know to remove the padding bits and thus can still recover the original message K

5. Now that the Great Council knows that there are spies for Ruby Cel among them, it has been decided that they need a simple and secure protocol for their private communications between each other.
Simple Simon, a new member of the council suggests the following protocol: user A sends a message block containing their own name, the message with their own name appended encrypted using B's public key, and B's Name. User B acknowledges receipt by sending back a message block containing B's Name, the message with B's name appended encrypted using A's public key, and A's name.
In notation:

1. A sends B the following [A, E_{B_public}([M,A]), B]
2. B acknowledges with [B, E_{A_public}([M,B]), A]

Cunning Kay objects to the protocol on the basis that there is too much redundancy, and it will waste Council funds to send messages with so much overhead. He proposes to revise the protocol to the following: user A sends a message block containing their own name, the message encrypted using B's public key, and B's Name. User B acknowledges receipt by sending back a message block containing B's Name, the message encrypted using A's public key, and A's name.
In notation:

1. A sends B the following [A, E_{B_public}(M), B]
2. B acknowledges with [B, E_{A_public}(M), A]

The Head of the Great Council is suspicious of Cunning Kay, believing him to be trying to make some kind of weakness in the protocol so that he can spy for Ruby Cel. However, the Great Council is currently in a cost cutting phase, so unless he can prove there is a weakness in Cunning Kay's revised protocol, the council will go ahead and implement it.
You are hired to check Cunning Kay's revision of the protocol for weaknesses.

How could a user C exploit the revised protocol to obtain M? Why is this not possible in the original protocol?

Solution:
In the revised protocol, if C is able to intercept A's original message to B, then C can send the following message to B: [C, E_{B_public}(M), B]. B will then acknowledge receipt to C with [B, E_{C_public}(M), C], and C will be able to obtain M.
In the original protocol, C cannot modify the part of the message that is already encrypted, and thus if C tries to send [C, E_{B_public}([M,A]), B], B can see that C has simply copied the message from A, and thus knows that it is not a valid message.