CS 161 **Spring 2016**

Homework #1

Nikhil Unni

1. Past-APF just meant that there's a difference between being able to access a message and understanding its meaning. Something that is secure should not be accessed by outside parties, but something that is private should not be able to be interpreted by third parties. What this implies about the design of cryptographic systems is that sometimes we cannot guard against parties snooping, but we can make sure that nothing cofidential is leaked.

2. (a) True
   (b) True
   (c) False
   (d) False
   (e) False

3. (a) Eve, the passive eavesdropper cannot do anything to find the encrypted messages, since she cannot reverse the one-way functions. Mallory, on the other hand, can intercept the initial email exchange of public keys, and just send them her own public key. From then on, all subsequent communication is locked with Mallory's public key, and she has the private key to decrypt them.

   (b) It is the same as part a – Eve cannot do anything, but Mallory can. Just in this case, Mallory hijacks g^x and g^y and can send her own g^m to both parties.

4. (a) Assuming that the plaintext is parsed from left to right, then we can have a username of "al", and a display name of "Alice!A". This results in a combined 10 bytes (with an exclamation mark separator), and the "!0" and random 4 bytes of padding will all be interpreted as garbage, and Alice will have administrator access.

   (b) Since we have C_0, and M_0 (where M_0 is just the first 16 bytes of the original plaintext : "alice!Alice X. J"), we can calculate $\text{AES}_K(IV + 0) = C_0 \oplus M_0$. Then, we just construct a new $M_0' =$ "alice!Alice!A000", and subsequently, $C_0' = \text{AES}_K(IV + 0) \oplus M_0'$. The following 16 bytes are just the original 16 bytes, since it doesn't depend on the previous 16 bytes. Therefore, we have:

   8452c68b712d36a0e0b900ff36dec332
   6df3d27be07554b6d25ce97b33ab70
   19892cfb1a95ac0e3ae97e3fb3bd9b88

   The Python program to calculate it:

```
first = 'alice!Alice X. J'
enc = '6df3d27be07554b6d25c0e96aa241b0a'
for i in range(0, len(enc), 2):
    split_up.append((enc[i] + "" + enc[i+1]).decode("hex"))
aes0 = [ord(first[i]) ^ ord(split_up[i]) for i in range(len(first))]
new_first = 'alice!Alice!A000'
c0 = [hex((ord(new_first[i]) ^ aes0[i])) for i in range(16)]
print ''.join(a[2:] for a in c0)
```

   (c) Say IV and M_0 are the original initialization vector and message respectively. When decrypting C_0, we calculate $\text{DEC}(C_0) = IV \oplus M_0$. So if we hijacked the IV so that we can encrypt our new message, $M_0' =$ "alice!A. Jones!A", and construct $IV' = IV \oplus M_0 \oplus M_0'$. Then, when the server receives the original C_0 with our constructed IV', we get, $M_{\text{received}} = \text{DEC}(C_0) \oplus IV' = (IV \oplus M_0) \oplus (IV \oplus M_0 \oplus M_0') = M_0'$. Therefore, we have:

   8d681dc517d8b37bc21b1960a9757031
   3aa64478061df9515a55a347d55cbdee

   The Python program to calculate it:

```
first = 'al1c3!A. Jones!0'
iv = '8d6845c541d8b37bc21b1960a975703f'
hex_iv = []
for i in range(0, len(iv), 2):
    hex_iv.append( int(iv[i]+iv[i+1] , 16) )
newPlain = 'alice!A. Jones!A'
new_iv = [hex_iv[i] ^ ord(first[i]) ^ ord(newPlain[i]) for i in range(16)]
print ''.join(hex(a)[2:] for a in new_iv)
```

5. (a) She just needs to calculate $B_1^{-k} * B_2$:

$$M = B_1^{-k} * B_2 = g^{-rk} * M * g^r k = M$$

(b) Say we have:

$$C_1 = (C_{1,1}, C_{1,2}) = (g^r, M_1 * g^{rk})$$
$$C_2 = (C_{2,1}, C_{2,2}) = (g^r, M_2 * g^{rk})$$

Then Eve can calculate $g^{rk} = M_1^{-1} * C_{1,2}$, by calculating the multiplicative inverse of M_1, since this is known. From that, she can find M_2 like:

$$M_2 = C_{2,2} * (g^{rk})^{-1}$$

(c) Say Mallory receives $C^* = (C_1^*, C_2^*) = (g^a, M * g^{ab})$ from Alice. Then, Mallory constructs a private key x, and public key g^x, and sends $(g^{ax}, M^x * g^{abx})$ it to Bob. Assuming that this is not a valid English sentence (and if it's not, Mallory can keep constructing new x values until it's so), Bob will send back to Mallory the unencrypted message : $g^{-abx} * M^x * g^{abx} = M^x = M^x$. Since Mallory knows x, she can compute $M = (M^x)^{x^{-1}}$.