Student: Uday Mahajan

UW NETID: udaym242

CSE 484 / CSE M 584 - Homework 2

This homework is focused on cryptography.

Overview

- **Due Date:** Wednesday, Nov 7, 2018, 4:30pm.
  - **Group or Individual:** Do this assignment as an **individual**. But you can talk with others in advance of actually doing the assignment.
- **How to Submit:** Submit a PDF to Canvas. Your assignment does not need to be entirely typed / developed with computer software. You could hand-write your assignment, and hand-draw some diagrams, and then submit a PDF scan of your hand-written assignment, just make sure it is *legible*.
- **Total Points:** 45 (across 9 questions)

Q1 (3 points).

What are the main concerns cryptographers have with the Encrypt-and-MAC method for combining a symmetric encryption scheme with a symmetric MAC to create a symmetric authenticated encryption scheme?

In Encrypt-and-MAC scheme, a MAC is produced based on the plaintext, and the plaintext is encrypted without the MAC. The plaintext's MAC and the ciphertext are sent together. MAC is deterministic, so same plaintext -> same MAC thus leading to privacy concerns.

Encrypt-and-MAC does not preserve confidentiality because two encryptions of the same message will have the same mac tag, revealing the fact that the messages are equal.

Q2 (2 points).

Consider a group of 30 people in a room who wish to be able to establish pairwise secure communications in the future. How many keys need to be exchanged in total:

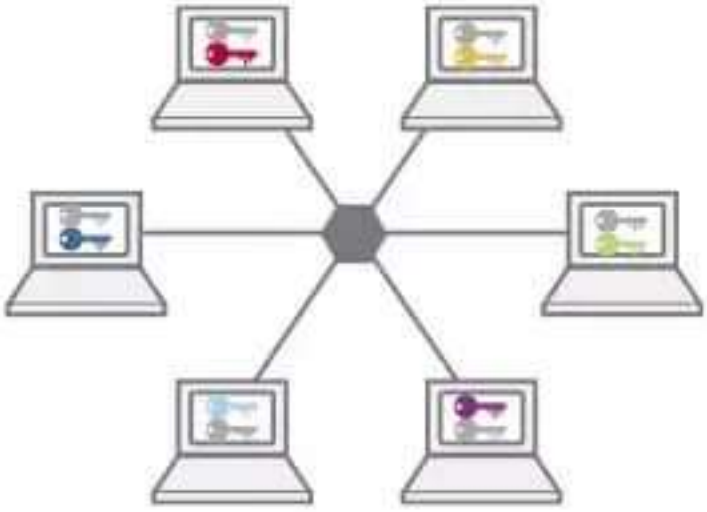
(a) Using symmetric cryptography?

In symmetric cryptography, two communicating parties have access to a

shared random string K, called the key. So, the total keys need to be exchanged is 30×29/2 = 435 keys. Below is a diagram of 6 people to convey the same idea.

(b) Using public key cryptography?

30 keys need to be exchanged. Each person will have one public and private key for private communication. Each of the person will share his or her public key for message encryption thus amounting to 30 keys being exchanged. The message is then decrypted using private key of the recipient which is not exchanged. Below is a diagram of 6 people to convey the same idea:



Q3 (5 points).

This message was encrypted with the RSA primitive, where N=33 and e=3. Decrypt it and submit the corresponding plaintext.

Tips: You are welcome to write a program to aid in the decryption, and you might want to compute the private decryption exponent d.

For this cryptogram 'A' is encoded as a 1 before encryption, 'B' as a 2, and so on.

Here is the cryptogram: 14 17 3 28 3 28 10 21 28 14 1 24 26 19 3 5 31 26 24 5 9 14 14 9 26 5 27 24 16 4 14 19 26 28 28 1 13 26 28 23 3 14 17 1 28 21 8 28 14 3 14 21 14 3 9 5 27 3 4 17 26 24 , 23 17 3 27 17 14 17 3 28 8 1 28 3 27 1 12 12 16 3 28 .

Ans. The cryptogram when decrypted gives the following result:

THIS IS JUST A REMINDER NOT TO ENCRYPT MESSAGES WITH A SUBSTITUTION CIPHER , WHICH THIS BASICALLY IS .

From given N= 33, we get p = 3 & q = 11. Therefore, phi(N) = (p-1)(q-1) = 20
We then compute d = e^(-1) % (phi(N)) = (3^(-1)) % 20 = 7

For ach integer "num" in the cryptogram, ae convert the it to text using num^(d)%N + 64 = num ^(7) %33 + 64 and get the above-mentioned result.

Please refer to the java program: http://www.udaymahajan.me/shared/files/484/Question3.java

Q4 (8 points).

The following question has you use RSA, but with larger values (but still not anywhere close to the size of the numbers one would use in a secure cryptographic protocol like TLS/SSL).

You may use a program that you write, Wolfram Alpha, or any other computer program to help you solve this problem.

For all of these, it is enough to just include your number in the answer, unless the question explicitly asks for additional detail.

Let p = 9497 and q =7187 and e = 3.

- **Compute N = p * q. What is N?**

- **Compute Phi(N) = (p-1)(q-1). What is Phi(N)?**
- **Verify that e is relatively prime to Phi(N). What method did you use to verify this?**
- **Compute d as the inverse of e modulo Phi(N). What is d?**
- **Encrypt the value P = 22446688 with the RSA primitive and the values for N and e above. Let C be the resulting ciphertext. What is C?**
- **Verify that you can decrypt C using d as the private exponent to get back P. What method did you use to verify this?**
- **Decrypt the value C' = 11335577 using the RSA primitive and your values for N and d above. Let P' be the resulting plaintext. What is P'?**
- **Verify that you can encrypt P' using e as the public exponent to get back C'. What method did you use to verify this?**

Please refer to the java program: http://www.udaymahajan.me/shared/files/484/Question4.java

p is 9497 and q is 7187
So, N = p * q = 9497 * 7187 = 68254939

Phi(N) = (p-1)*(q-1) = 68238256.
e = 3

d = e^(-1) % phi(N) = 45492171

To verify that e and phi(N) are indeed relatively prime we can verify if the gcd(e, phi(N)) results in 1. As seen in the java program, it returns 1.

To get encrypted cipher text C from given text P we compute P ^(e) % N.
To get decrypted cipher text P from given cipher C we computer C ^(d) % N.

Given text P = 22446688, I got ciphertext C = 23081176 by using the encryption method stated above. I was able to get the decrypted ciphertext P = 22446688 back using decryption method from above.

Similarly, given the encrypted text C' = 11335577, I was able to get the decrypted text P' = 35654065 and also go back to encrypted text C' = 11335577.

Q5 (5 points). Suppose you, as an attacker, observe the following 32-byte (3-block) ciphertext C1 (in hex)

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  03
46 64 DC 06 97 BB FE 69 33 07 15 07 9B A6 C2  3D
2B 84 DE 4F 90 8D 7D 34 AA CE 96 8B 64 F3 DF  75
```

and the following 32-byte (3-block) ciphertext C2 (also in hex)

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  03
55 7A C0 17 90 B1 FD 74 3C 18 09 0C DE 90 C3  21
2F 98 D4 4B 8D 99 63 28 B6 9C F5 C3 34 F7 C5  62
```

Suppose you know these ciphertexts were generated using CTR mode, where the first block of the ciphertext is the initial counter value for the encryption. You also know that the plaintext P1 corresponding to C1 is

```
43 72 79 70 74 6F 67 72 61 70 68 79 20 43 72  79
70 74 6F 67 72 61 70 68 79 20 43 72 79 70 74  6F
```

Compute the plaintext P2 corresponding to the ciphertext C2. Submit P2 as your response, using the same formatting as above (in hex, with a space between each byte).

Please refer to java program: www.udaymahajan.me/shared/files/484/Question5.java

We know that the CTR mode of cryptography, we encrypt the P1 by XORing it with keystream we get out of the block cipher.

As seen in the above .java code, we can get the keystream k by doing P1 XOR C1 due to properties of XOR.

$KS \oplus P1 = C1$
$KS \oplus P1 \oplus P1 = C1 \oplus P1$ (XOR both sides with P1)
$KS = C1 \oplus P1$ ($X \oplus X = 1$)

Once we get the keystreams of the blocks, we can XOR the given C2 blocks with those keystreams to get P2 blocks. Now we have the keystream and can compute the plaintext P2 as follows:

KS ⊕ P2 = C2
=> KS ⊕ P2 ⊕ P2 = C2 ⊕ P2 (XOR both sides with P2)
=> KS = C2 ⊕ P2 (X ⊕ X = 1)
=> KS ⊕ C2 = C2 ⊕ P2 ⊕ C2 (XOR both sides with C2)

=> KS ⊕ C2 = P2 (X ⊕ X = 1)

We must do this separately for both the blocks as each block will have different keystreams due to incrementation of initial counter.

The resulting P2 is given below:

```
50  6c  65  61  73  65  64  6f  6e  6f  74  72  65  75  73  65
74  68  65  63  6f  75  6e  74  65  72  20  3a  29  74  6e  78
```

If we convert the above P1 in hex to ASCII, we get the following message:
Pleasedonotreusethecounter :)tnx

Q6 (5 points).

Consider an insecure version of SSH that uses ECB mode for encryption. Whenever a user types a key into the ssh client, that key is immediately encrypted and sent over the wire to the server. This immediate encrypt-after-key-press procedure is what enables the interactivity of a remote shell. Now consider the following sequence of plaintext packets (written in hex):

P1 = 6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII l
P2 = 73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII s
P3 = 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII space
P4 = 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII *
P5 = 2D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII -
P6 = 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII f
P7 = 72 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII r
P8 = 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII o
P9 = 6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII m
P10 = 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 // ASCII <enter>

This corresponds to a user typing “ls *-from<enter>” into their ssh client.

Suppose an attacker knows what the user is typing via some out-of-band channel (e.g., shoulder surfing) and also eavesdrops on this communications and intercepts the corresponding ciphertexts:

C1 = 4E B6 48 B2 E0 BE A5 B1 21 2F 07 54 DF CF A4 39
C2 = 11 70 78 65 88 89 06 62 82 0C 0A 6A 55 6F 87 46
C3 = EF 7F 1F 25 3E 99 98 8D 1A FD BE 7A D9 D6 ED 7E
C4 = 5B 40 2B 18 0B 94 E8 13 DA F3 DE 21 A0 27 2E C4
C5 = 93 80 19 1F 06 B4 4B 19 9D 70 86 28 34 12 26 DC
C6 = 68 74 EB 1B 16 5F 70 45 05 29 B9 66 0A CC D3 6C
C7 = 56 E8 77 E1 7E BF 01 19 27 87 03 FE E1 1D 65 A8
C8 = 9D 37 51 F0 68 C8 F7 BA 44 B2 E9 5C 09 94 1D 5A
C9 = 62 30 38 8F A4 D7 C1 56 68 88 CE 2C 29 2D F5 23
C10 = D5 89 74 7E 45 89 08 FA 5B 63 98 42 E6 B2 31 85

The attacker can now inject messages into the communications channel from the client to the server. One thing an attacker might try to do: generate a sequence of ciphertext packets that, when decrypted, are interpreted as "rm -rf *<enter>" on the server. Give such a sequence of ciphertext packets in your answer.

The attacker will use the following sequence to inject "rm -rf *<enter>" to the communication channel:
Sequence (based on original cipher character naming) : C7, C9, C3, C5, C7, C6, C3, C4, C10

Therefore,

C1 = 56 E8 77 E1 7E BF 01 19 27 87 03 FE E1 1D 65 A8 // Original C7 - ASCII r
C2 = 62 30 38 8F A4 D7 C1 56 68 88 CE 2C 29 2D F5 23 // Original C9 - ASCII m
C3 = EF 7F 1F 25 3E 99 98 8D 1A FD BE 7A D9 D6 ED 7E // Original C3 - ASCII " "
C4 = 93 80 19 1F 06 B4 4B 19 9D 70 86 28 34 12 26 DC // Original C5 - ASCII "-"
C5 = 56 E8 77 E1 7E BF 01 19 27 87 03 FE E1 1D 65 A8 // Original C7 - ASCII r
C6 = 68 74 EB 1B 16 5F 70 45 05 29 B9 66 0A CC D3 6C // Original C6 - ASCII f
C7 = EF 7F 1F 25 3E 99 98 8D 1A FD BE 7A D9 D6 ED 7E // Original C3 - ASCII " "
C9 = 5B 40 2B 18 0B 94 E8 13 DA F3 DE 21 A0 27 2E C4 // Original C4 - ASCII "*"
C9 = D5 89 74 7E 45 89 08 FA 5B 63 98 42 E6 B2 31 85 // Original C10 "<enter>"

Q7 (3 points). Consider a Diffie-Hellman key exchange with p=29 and g=2. Suppose that Alice picks x=3 and Bob picks y=5. What will each party send to the other, and what shared key will they agree on? Show your work.

Alice will send the key k1= g^(x) mod p = 2^(3) % 29 = **8**

Bob will send the key k2= g^(y) mod p = 2^(3) % 29 = **3**

Shared key K: g^(xy) mod p = 2^(15) % 29 = **27**

Q8 (5 points).

The goal of this task is to give you a better understanding of Certificate Authorities (CA) and certificates.

Look at the CAs certificates that your computer trusts.

- **Mac: Spotlight search 'Keychain Access'. Open the Keychain Access app and inspect the "System Roots" keychain.**
- **Windows: Control Panel -> Search 'Internet Options' -> Content -> Certificates (and then look at the various tabs)**

Answer these questions:

1. **How many root CA certificates does your computer have?**

   I have total 48 root CA certificates on my Windows laptop.

2. **What is something that you found interesting from looking at the root CA certificates?**
   I found that I am not allowed to delete these root CA certificates.

3. **Go to google.com using your favorite browser and find a way to look at the certificates for google.com. List the chain of certificates your browser sees.**

   Google.com sees the certificate from Global sign that Ensures the identity of a remote computer, proves your identity to a remote computer, ensures software came from software publisher, protects software from alteration after publication and protects e-mail messages. Following is the chain of certification from Global sign to Google on Chrome browser.

4. **What is a possible risk of trusting a CA?**

   If we add a certificate to Trusted Root Certificate Authorities, then any certificate signed by that certificate will be trusted by your machine. So, if one of the certificates signed by that authority gets compromised, then all the machines where it is installed become vulnerable to man in the middle attack.

Q9 (9 points).

For this task, the goal is to give you experience with sending encrypted emails. To successfully complete this task, you will need to set up your email client and send/receive an encrypted email to/from the TAs. For this assignment, you can reach your super-secret agent TA Charizard at charizard.thehacker@gmail.com.

Using the email account that you associated with your key above (you'll get an error back if your email account and key don't match), send an email to the TA in this format:

To: charizard.thehacker@gmail.com

Subject: [CSE 484] Encrypted email

Content: Whatever secret message you'd like to send us :)

Attachment: (Select your key from 'Display Keys' on the Mailvelope site, and export the public key only. Download _pub.asc and attach it.)

1. **The email address you used :**
   udaym242@uw.edu
2. **Secret value provided by the TA :**

78d665c4ee31d189d84150b804de8833103af4f443191cafae4c1c90e2653bdd

3. Answers to short answer questions

a. Does this process (PGP encryption) involve the use of symmetric or asymmetric encryption or both?

Both. PGP combines symmetric-key encryption and public-key encryption. It uses Asymmetric encryption to encrypt a symmetric encryption key. The symmetric encryption key encrypts the data.

b. We recommended a browser extension for ease of use, but what are the security risks of enabling this browser extension? (Hint: what permissions did the extension ask for during install?)

The browser extension can essentially read all the content of the websites I visit including sensitive information in emails. If the developer of the extension acts like an adversary to steal the data, I might lose my privacy at the expense of encryption