COS 116
The Computational Universe
Laboratory 10: Cryptography

In this lab you'll learn several methods of encrypting messages to maintain confidentiality, and you'll learn how some of these encryption methods can be broken.

In Parts 1,2,3, and 5 of this lab, you'll be working with a partner. In Parts 1 and 3, one partner will encrypt a secret message that the other partner will decrypt. In Parts 2 and 5, you and your partner will try to break another group's encrypted message, and that group will try to break yours.

Even though you'll be working with a partner, please write your lab report on your own.

Hand in your lab report at the beginning of lecture on Tuesday, April 24. Include responses to questions printed in bold. (Number them by Part and Step.)

Part 1: Substitution Cipher

You need a partner for this part.

A substitution cipher is one of the simplest encryption methods. It's only a small step up from the Caesar's Cipher discussed in class, and it's the one that's used in the "Cryptogram" puzzles in the newspaper. It works as follows:

Alice, the sender, and Bob, the receiver agree in advance on a table that maps each letter of the alphabet to a different letter of the alphabet. This table is the secret key that only Alice and Bob are supposed to know. Here is an example of such a table:

Letter	Replace with
A	Z
B	G
C	S
D	L
E	Y
F	_
G	H
H	C
I	R
J	T
K	X
L	Q
M	B
N	U
O	F
P	E
Q	W
R	K
S	D
T	O
U	N
V	J
W	A
X	V
Y	P
Z	M
_	I

Note that in this table, we have a 27^{th} letter "_" that represents a space.

Now suppose that Alice wants to send Bob the secret message:

MEET AT NOON

Alice can encrypt the message by replacing each letter of the message with the appropriate letter from the "Replace with" column of the secret table. The encryption of the message would be:

BYYOIZOIUFFU

To decrypt the message, Bob, who knows the same secret table as Alice, can simply do the substitution in reverse.

1. **With your partner, agree on a substitution table like the one above that maps each letter of the alphabet to another letter of the alphabet. Make sure your table has a 27th entry for the space character "_". Record your table in your lab report.**

2. **Decide with your partner which one of you will be the sender and which one will be receiver.**

3. **If you are the sender, do the following:**

    a. **Devise a secret message, but do not reveal it to your partner. To make your encrypted message easier to break in Part 2, make the message you choose start with the text:**

        THIS IS A SECRET MESSAGE

        Your message should be between 80 and 120 letters long including spaces and the required starting text. Use all capital letters and don't use any punctuation.

    b. **Encrypt the message using the substitution table that you agreed on with your partner. Write the ciphertext of the message neatly and clearly on a separate sheet of paper, and then give the paper to your partner.**

4. **If you are the receiver, wait for your partner to give an encrypted message. When you receive it, use the substitution table that you agreed on with your partner to decrypt the message. One you've decrypted the message, confirm with your partner that you've decrypted it correctly. Save the sheet of paper with the ciphertext of the message; you'll need it for Part 2.**

5. **Record the original message text and the ciphertext, in your lab report.**

Part 2: Breaking a Substitution Cipher

You need a partner for this part.

1. **Find another group that has finished Part 2, and give them the sheet of paper with the ciphertext of the message that you sent in Part 1. In return, you'll get the ciphertext of their message.**

2. **Work with your partner to try to break the other group's message (i.e. try to decode it without knowing the secret table that the other group used).** Here are some tips that may help:

   - Trying to blindly guess the other group's substitution table at random isn't going to work. There are 27!, or more than 10^{28}, possible substitution tables. Even a computer can't try that many possibilities.

   - The fact that all messages begin with "THIS IS A SECRET MESSAGE" should be very helpful. It should allow you to learn a number of the rows of the substitution table that the other group used. You might think that having every message start with the same prefix is unrealistic, but it's not as unrealistic as you might think. In the real world, messages, especially those exchanged by computers, often have standard formats and often begin with well-known sequences of bytes.

   - The frequency that letters appear in the encrypted message might give something away. The letters that appear most frequently in the ciphertext of the message may very well correspond to letters that appear most frequently in English text. You can get a list of letter frequencies at:

     http://en.wikipedia.org/wiki/Letter_frequencies

   - Once you have learned some of the rows of the substitution table, especially the mapping for the space character, you can start to deduce words of the message. Correctly guessing words should, in turn allow, you to deduce the mappings for more letters.

DON'T SPEND MORE THAN AROUND 30 MINUTES ON THIS PART OF THE LAB. THERE ARE 3 MORE PARTS THAT YOU HAVE TO GET TO. IF YOU HAVEN'T DECODED THE MESSAGE AFTER 30 MINUTES, MOVE ON.

3. **Record the ciphertext of the other group's message in your lab report. If you successfully decoded the message, include the decoded version in your lab report. If you haven't, include your progress in your lab report.**

4. **Explain how you went about decoding the message. Which strategies were most effective?**

Part 3: Pseudorandom Generator

You need a partner for this part.

A pseudorandom generator is an algorithm that takes a secret key as input and outputs a "random-looking" sequence of numbers called a *keystream*. The keystream is actually not random at all — given the same secret key, a pseudorandom generator will always produce the same keystream. However, there is no known efficient way for someone who does not know the secret key to predict what the keystream will be. Thus, from his or her perspective, the keystream might as well be truly random.

Alice and Bob can use a pseudorandom generator in manner similar to the way they would use a one-time pad. Instead of agreeing on an entire one-time pad in advance, Alice and Bob can simply agree on a relatively short secret key. Then, when Alice wants to send an encrypted message to Bob, she can input the secret key into a pseudorandom generator and get a keystream. She can then add that keystream to the message just as she would add a one-time pad to the message and then send the resulting ciphertext to Bob. To decrypt the message, Bob can enter the secret key into his own pseudorandom generator and get back the same keystream that Alice used to encrypt the message. Finally, Bob can subtract that keystream from the ciphertext to produce the original message.

1. **With your partner, agree on a secret key. The secret key can be made up of letters and numbers and can be as short as the passwords that you use to log into Web sites.**

2. **Record your secret key in your lab report.**

3. **Decide with your partner which one of you will be the sender and which one will be receiver.**

4. **If you are the sender, do the following:**

    a. **Devise a secret message, but do not reveal it to your partner. Your message should be less than 80 letters. Use all capital letters and don't use any punctuation. The message does not need to start with any particular text.**

    b. **Go to the following web site:**

    http://www.cs.princeton.edu/courses/archive/spring07/cos116/lab10/prg.php

    c. **Enter you secret key into the text box on the site and click 'Generate.' The site will use a pseudorandom generator to produce a keystream. The keystream will consist of a string of random numbers. Each number will be between 0 and 26.**

d. Encrypt your secret message with the keystream. Do the following:

i. Represent each letter of your message as a number: A is 0, B is 1, C is 2, Z is 25, and the space character is 26, etc.

ii. Add each letter of your message to the corresponding number in the keystream and then take the result mod 27. The resulting string of numbers is the ciphertext.

For example, if your message were:

MEET AT NOON

You would first represent it as numbers as follows:

```
12 4 4 19 26 0 19 26 13 14 14 13
```

Then, if your keystream were:

```
3 5 10 25 16 13 7 6 14 14 22 23
```

you would add it to your message mod 27 to get:

```
15 9 14 17 15 13 26 5 0 1 9 9
```

This would be your ciphertext. Notice that, because the result of the addition is being taken mod 27, sums that are greater than or equal to 27 "wrap around."

e. Write the ciphertext of the message neatly and clearly on a separate sheet of paper, and then give the paper to your partner.

5. If you are the receiver, wait for your partner to give an encrypted message.

a. Go to the following web site:

http://www.cs.princeton.edu/courses/archive/spring07/cos116/lab10/prg.php

b. Enter you secret key into the text box on the site and click 'Generate.' The site will use a pseudorandom generator to produce a keystream. The keystream will consist of a string of random numbers. Each number will be between 0 and 26.

c. **Decrypt the secret message with the keystream. Do the following:**

   i. **From each number of the ciphertext, subtract the corresponding number in the keystream and take the result mod 27. If the result is negative, add 27 to it.**

   ii. **Convert the numbers back to letters. Remember: A is 0, B is 1, C is 2, Z is 25, and the space character is 26, etc.**

d. **One you've decrypted the message, confirm with your partner that you've decrypted it correctly.**

6. **Record the original message text and the ciphertext, in your lab report.**

Part 4: Diffie-Hellman

You don't need a partner for this part.

WARNING: There is a bit of unavoidable math below. Please read it; if you have questions, don't hesitate to ask.

Diffie-Hellman is public-key cryptosystem that allows two perfect strangers to establish a shared secret key without ever agreeing on anything in advance. Here's how it works.

As in other public-key cryptosystems, each person has two keys: a *private key* that only he or she knows and a *public key* that everyone knows. For Diffie-Hellman, a person's *private* key is simply a random number – call it x. A person's *public* key is equal to $g^x \bmod p$ where x is the private key, p is a large prime number and g is a small, but carefully chosen number.

Suppose that Alice wants to send a message to Bob, but that Alice and Bob are strangers and haven't agreed on a secret key in advance. Also, suppose that Alice's private key is a and her public key is $g^a \bmod p$, and suppose that Bob's private key is b and his public key is $g^b \bmod p$.

To send an encrypted message, Alice takes Bob's public key, raises it to the a power (Alice's private key), and then takes the result $\bmod p$. This gives her:

$$(g^b \bmod p)^a \bmod p$$

which is the same as

$$g^{ab} \bmod p$$

Alice can then enter this value $g^{ab} \bmod p$ into a pseudorandom generator to get a keystream that she can use to encrypt the message using same method that you used in Part 3.

Now, to decrypt the message from Alice, Bob uses a similar procedure. He takes Alice's public key, raises it to the b power (Bob's private key), and then takes the result $\bmod p$. This gives him:

$$(g^a \bmod p)^b \bmod p$$

which is the same as

$$g^{ab} \bmod p$$

Notice that Bob has just arrived at the same value as Alice - $g^{ab} \bmod p$ - without ever agreeing on anything with her! Bob can then enter this value into a pseudorandom generator to get a keystream that he can use to decrypt the message using same method that you used in Part 3.

Now that you see that Diffie-Hellman works, the next question is: why is it secure? The security of Diffie-Hellman, like the security of RSA that you heard about in lecture, relies on a problem that is easy to create, but seemingly difficult to solve. This problem is known as the "discrete logarithm problem." If you are just given g^x and you wanted to get x, you could just take the logarithm of g^x. But, if you are given $g^x \bmod p$ and you wanted to get x, there is no known efficient way to do it. That extra $\bmod p$ makes a difference.

1. **Go to the following web site and log in:**

   http://www.cs.princeton.edu/courses/archive/spring07/cos116/lab10/dh_keys.php

   On the web site, you will see your own Diffie-Hellman *private* key along with everyone's *public* keys.

2. **Pick a person in the class to send an encrypted message to and write down their public key. Try to pick someone who is already done with Part 3 of the lab and who has not yet received an encrypted message in Part 4.**

3. **Record the person that you chose in your lab report.**

4. **Pick a secret message less than 80 letters long to send to them. Use all capital letters and don't use any punctuation. The message does not need to start with any particular text. Record the message in your lab report.**

5. **Take the recipient's *public* key and raise it to the power of your *private* key and take the result** $\bmod p$**. In your case,** $p = 51683$**.** In other words, if the recipient's public key is y and your private key is x, then you want to calculate:

$$y^x \bmod 51683$$

The numbers involved in these calculations are too big for an ordinary calculator to handle.

To perform the calculation, use the Big Number Calculator at:

http://world.std.com/~reinhold/BigNumCalc.html

Here's how to use it:

   a. **Enter 51683 into the middle of the three text boxes (the one which says y=) and click on the "=" button.**

   b. **Then, click on the "SetMod" button. Once you do this, the bottom text box should say: "16 bits, base 10, modulo 51683"**

   c. **Enter the recipient's public key into the middle text box and click on the "x**y" button.**

   d. **Enter your private key into the middle text box and click on the "=" button. The result of the calculation appears in the top text box.**

6. **Encrypt your secret message using the process described in Part 3 Step 4 where the secret key is the value that you just calculated.**

7. **Write the ciphertext of the message *along with your name* neatly and clearly on a separate sheet of paper and hand the sheet of paper to the recipient.**

8. **Record the ciphertext in your lab report.**

9. **If you receive an encrypted message from someone else, do the following steps.**

10. **Go back to:**

    http://www.cs.princeton.edu/courses/archive/spring07/cos116/lab10/dh_keys.php

11. **The sender's name should be written on the sheet of paper that you received. Look up the sender's public key on the web site and write it down.**

12. **Record the sender's name in your lab report**

13. **Take the sender's *public* key and raise it to the power of your *private* key and take the result** $\bmod p$**. In your case,** $p = 51683$**.** In other words, if the sender's public key is y and your private key is x, then you want to calculate:

$$y^x \bmod 51683$$

    Use the Big Number Calculator to calculate this.

14. **Decrypt the message that you received using the process described in Part 3 Step 5 where the secret key is the value that you just calculated.**

15. **Record both the encrypted and decrypted versions of the message in your lab report.**

16. **Ask the sender whether you decrypted the message successfully and record whether or not you did in your lab report.**

Part 5: Why Reusing a One-time Pad Leads to Disaster

You need a partner for this part.

In this part, you will encrypt a new message under the same secret key that you used in Part 3. Since entering the same secret key into a pseudorandom generator produces the same keystream every time, you will end up encrypting two different messages with the same keystream. This has the same effect as using the same one-time pad to encrypt two different messages. You heard in lecture that reusing a one-time pad is a bad idea; now you will see why.

1. **Devise a new secret message that is different from the one you used in Part 3. Your message should be less than 80 letters. Use all capital letters and don't use any punctuation. The message does not need to start with any particular text.**

2. **Follow the instructions in Part 3 Step 4 to encrypt the message. USE THE SAME SECRET KEY THAT YOU USED PART 3.**

3. **Write the ciphertext of the new message neatly and clearly on a separate sheet of paper**

4. **Record the new ciphertext in your lab report**

5. **Find another group that has gotten to Part 5, and give them the following:**

   - **The ciphertext of the message you sent in Part 3**
   - **The ciphertext of the new message that you just encrypted**
   - **The *plaintext* of the message (i.e. the original unencrypted message) that you sent in Part 3**

   You will receive the other group's messages in return. Record them in your lab report.

Armed with the ciphertext of both of the other group's messages along with the plaintext of one of the two messages, you should be able to deduce the keystream that was used to encrypt both messages. Once you know the keystream, you can decrypt the second message.

6. **Using the plaintext and ciphertext that you received from the other group, deduce the keystream that the other group used. Think about how you can use both the plaintext and ciphertext that you know. Once you've deduced the keystream, record it in your lab report.**

7. **Now that you know the keystream, decrypt the other group's second message and record it in your lab report.**

8. **It might seem unrealistic that you were told the plaintext of one of the other group's messages, but there are actually many situations in which an attacker sees a message in encrypted form, and then later discovers the plaintext that corresponds to it (without decrypting the message). What is one such situation?**