Introduction to Computer Security – G6077

This lab covers

1) Understand caesar, Xor and Affine ciphers
2) Implement and test ciphers in a programming language
3) Compare and analyse different ciphers implementations

Contents

Tasks Caesar cipher

Task 1

Without searching on the Internet, can you demonstrate your understanding of Caesar cipher in the form of writing your own algorithm/pseudocode.

Task 2

Implement your algorithm of task 1 using any programming language. Test it by different values to check the logic is correct.

Task 3

Compare your solution with the one provided on Canvas. Which one is more efficient and why?

Affine Cipher

Read the document titled Affine cipher provide in the week03 lab resources and then complete the following tasks once you understood the Affine cipher.

Task 1

Without searching on the Internet, can you demonstrate your understanding of Affine cipher in the form of writing your own algorithm (logical steps).

Task 2

Implement your algorithm of task 1 using any programming language. Test it by different values to check the logic is correct.

Task 3

Compare your solutions with the one provided on Canvas for these ciphers.

XOR Cipher

Read the following article to understand XOR cipher and then complete tasks.
Article 1) https://www.hackthis.co.uk/articles/the-xor-cipher

Task 1

Without searching on the Internet, can you demonstrate your understanding of Affine cipher in the form of writing your own algorithm (logical steps).

Task 2

Implement your algorithm of task 1 using any programming language. Make sure that you test it for different values.

Task 3

Compare your solutions with the one provided on Canvas for these ciphers.

Cipher challenges

The link provided at the end of this paragraph is a very useful resource to check your understanding of the different ciphers that we studied in lecture. It has many examples and challenges. You must complete at-least one challenge for each of the ciphers that we have studied so far. For most of the ciphers, a brief video is also provided by the author, who is a well-known person in the area of computer security, Prof. Bill Buchanan.

https://asecuritysite.com/challenges

To verify and check the answer of any of the challenge, you should use the CrypTool listed below. You should also double check your answer with a friend.

CrypTool – Online tool

CrypTool is a software used to encrypt and decrypt messages. It provides an exciting insight into variety of ciphers, coding methods and analysis tools. It can be installed on machines. It has an online version which runs in the browser called CrypTool-Online (CTO). The CTO can be accessed by clicking on https://www.cryptool.org/en/cto-ciphers .CTO has most of the ciphers that we have covered in the course.

Task- Your own cipher

Write your own cipher that may involve substitution and transposition.

Write logical steps of your cipher, then use any programming language and implement your algorithm. Don't forget to test it.

[This task will test your understandings of the important mechanisms involved in different ciphers. You will notice in complex ciphers like DES and 3DES, that substitution and transposition are at their cores.]

More explanation on Caesar cipher

The idea is to shift the letters; it is sometimes called Shift cipher. It operates with letters not bytes not bits, it is a historic cipher. It should be noted that the modern crypto systems are working with bytes and numbers.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

If I shift letters by four characters the encryption would look like this.

It grabs A and turns it into A plus four so A becomes E.

Then we grab B from plaintext and turn it into F.

Then we grabs C and turn it into G and so on and so on until it reach U and U turn into Y then we reach V and V turn into Z. So now we reach W but we don't have a letter behind Z; Z is where our alphabet ends.

So what do we do.

You might be able to figure it out it is not so hard but let us see maths and logic behind it. It has to do something with the modulo operator.

So how modulo operator works if we have let's say 20 modulo 26 What do we get?

We get 20 right because we want the remainder of division and when we divide 20 by twenty six. The remainder is 20 right.

What if I do?

Twenty seven modulo twenty six. The result will be 1.

Because that is the rest after division.

And if you look at the alphabet it ends with Z and starts with A.

So if you add a number to every letter it look like as listed above. Can a modulo operator give you zero? If you are not sure, can you do a little research on it.

Well in our case we shift by 4.So accordingly

23 modulo 26 = ? But wait a minute we need to slightly modify this

$$E(x) = (x + k) \bmod 26$$

In above equation, x is the number/letter of plaint text in this case x is 23 and k is the key, in this case k is 4. The total number of letters are 26 that is why we use 26 mod.

Thus for Z, the equation will looks like:

$E(22) = (22+4) \bmod 26 = 0$ and zero is A which is exactly four space away from W. You can calculate for X, Y and Z.

But in general case you can shift by any number and the number by which you are shifting is the **key**.

So now we got the ciphertext and we want to decrypt it into plaint text into our original message.

If you want. So how do we do that. Once again it is not so hard.

You need to think about how we changed the original character.

We added K to it.

So to get the origin message we just subtract the key but we also need to do modulo 26. The formula will look like:

$$D(y) = (y - k) \bmod 26$$

// use this calculator to find mod values, check for negative values

So we get our origin or character and that is basically how Caesar cipher works.

Now do you think this cipher is safe.

Noope ☹

First you can do frequency analysis to break it like we looked at an example in lecture, this is even easier and we will not need frequency analysis to break it.

Would brute force attack work. Ans: Yes.

Simple brute force attack would break this cipher. Why?

That is true but now I want you to notice one thing.

What is the difference between K equal to 27 and K equal to 1.

The answer is there, absolutely no difference.

It is the same cipher.

Why because twenty seven mod 26 is 1 and 1 mod 26 is also 1.

So we are actually shifting by one both times.

So you need to go through only 25 options in order for you to find the key and that is not a lot of work.

So the algorithm is not secure.

But it is one of the simplest one.

More on substitution cipher

Substitution cipher operates with letters. At the time it was created, nobody knew what a binary system is. At that time, there were only words and letters. Do you know what substitution is?

It may seem complicated but it means replacement.

Therefore, a substitution cipher takes a character and replace it by a different one.

For example, the encryption can look like this a is replaced by g, b is replaced by t, c is replaced by n and so on.

a ----------> g
b ----------> t
c ----------> n

Table 1

So now, for example a text like abc will be represented as gtn ciphertext. What will be baacac equal to? Ans: ----------

So now do you think this cipher is secure.

The right answer is No. This is not a secure cipher.

Then there is the question how would you break it.

First approach:

So the first approach of how to break the cipher is through **brute force**.

So imagine that we are attackers and we don't know table 1.

We do not know which character replaces which, and how many options are there. We need to figure out the table because we need the key in order for us to be able to decrypt the ciphertext.

If you have already learned something about discrete mathematics you know that there is a concept called, permutation and you probably know the answer already.

We are trying to find a replacement for a letter.

So how many options do we have?

Since we do not know anything about this cipher, it can be any letter and we have 26 letters I guess.

So I write 26 then we need to know what character is replacing b.

And we already have one character that replaces a.

So we are picking a character but this time there are only 25 characters available because we already picked one that replaces a.

And we cannot replace b by the same character.

Then why don't we pick character replacing c.

We have twenty-four options and we go all the way down to 1.

So that is a factorial all of twenty five.

So we have factorial of 25 options.

How key could look like... 26! * 25! * 24!..... 1!

That is a huge number.

It will take a good amount of processing time to calculate this number.

So brute force is a difficult choice here because there are too many possible keys.

Second approach:

The second approach maybe a bit smarter. In the second approach we can use frequency analysis, what does frequency analysis means?

Well the problem with this cipher is that when we take a character we always turn it into the same one.

So for example, A will be replaced as a G and are safer.

But the problem is that some letters in plain text are more frequent than others for example the letter E is the most common letter in English almost 13 percent of all others and the next common letter is the letter T with about 9 percent.

Well we can just look at the ciphertext and we count the frequency of characters in it and based on that information we can figure out the key the transformation table if we want.

So if I add here for example that E will be replaced by D and in this ciphertext if we find a letter that have a frequency of about 13 percent it is probably in plain text.

So that is basically how we can break these replacement encryption.

And it works because of static transformation because they will always map to G.

If you are the good guy that is creating ciphers and securing data you have a problem because even though the attacker would for example fail in one approach he can try a

different one and if one approach breaks your cipher your cipher is unsecured. Even though all the other attacks were unsuccessful.

So now, I want to talk about the approach this attacker can take.

And the first one is the simple brute force as you saw before where the attacker just simply tries all possible keys. And with each key it looks at the decrypted text and checks whether it is a valid text.

On the other hand there is a **monolithic attack**. And what that means.

Well an attacker really looks inside the encryption and decryption functions and tries to figure out some smarter way of breaking it. So for example like we did with the frequency of characters in the text that was a really monolithic attack but unfortunately it does not end here.

Another type is for example **social engineering**. You know this term from lecture resources. For example if I need a password of some person to their e-mail I can call them that I am from the I.T. Department and we lost their password or something like that and that I need their password and these sorts of attacks are very common.

And then it kind of doesn't matter how strong your crypto function is when you give away your password.

And the last one is the most moderate and it is all **implementation attack**. I will explain this one by an example.

Imagine that you have a credit card you know that does have a microprocessor inside the card. Something like AD51 and this microprocessor runs the encryption algorithm. And also on the card is somewhere stored key and this key is much hidden. You cannot access it.

But what you do you take some machine that measures power consumption and you trigger the card. So you basically say that I am ATM start encrypting and you measure the power consumption and this power consumption has something to do with the key. This key is then channel into a memory and can be easily retrieved at any time from memory.

Thankfully, since we know about this type of attack we can also defend against it.

So your credit card is safe but this was just one example of implementation attack.