DES – written later

ECB – Electronic Code Book mode

CBC mode – Cipher Block Chaining mode

Numbers, letters, Binary code

I intend to keep on using these examples with shifting of alphabets or the

substitution of alphabets just for a little longer.

I have a couple more things that I want to show you. But before we get there I just want to give

the understanding of a small technical point that's going to be useful for me to show you the rest. This isn't terribly important for your understanding of cryptography, it just helps to get through the rest.

Now in class we looked at this example. This was an alphabet shift cipher that I showed you and we found out that at least one of the keys that you could have found that was by shifting my inner circle nearly all the way around until it got

to there. Yeah? And because I always shifted it around in that direction we

called this a shift of 25. Just as if it had been the Caesar cipher we would call that one two three a shift of three. Yeah? So we have a shift of three as well in

that case. I'd just like to introduce the idea that well we could have a

different name for this because if I shift something to there I could just as

well call this, I could mark the a, take the a as my point of reference and

say that this is a shift to z. No? Same thing I'm just calling it that instead

of calling it 25. Same thing here that in principle a shift to, a shift of three

is a shift to d. Okay? That'll make things a little easier in what comes. Note

also that this means that when I'm doing the Caesar cipher as I'm doing here we

could say that I'm sort of adding the letter d to the letter a as I encrypt.

I'm also adding the letter d to letter b to get e and so on. So in a way it's

adding a letter to a letter. And this makes perfect sense to those of you who

are used to working with data representations of letters because there

are all numbers underneath aren't they? So that works of course until I get

around to here because if I get to w then I can't add, oh sorry, x. If I get to x

then I can't add a d because there's nothing further than z. So you see what

we do is we start again. So in data technical terms we can say we are adding

the 26 letters of the alphabet and then if it gets to be too much we're taking a

mod 26 operation on that, i.e. dividing by 26 and taking the remainder. And that's

what makes it circular like that. So point being we're kind of

adding one letter to another and if you follow that then you'll easily be able

to follow the remainder. One more little point though. We did this thing, this kind

of thing in class where I invented a message. I can't now remember exactly

what it was but let's go with that. And then I had students in class help me to

find, I do seem to remember it was something like that and then it went to

that and then maybe it was like that. Okay and this was our key. And we called

the operation to encrypt XOR. But in a way this is exactly the same as what I'm

going to do with letters, or what I've been doing with letters, because I'm

adding one letter to another and doing mod 26 on that. Well this is an alphabet

with only two positions, 0 and 1. I could have said A and B. But note how the XOR

is really the same as adding and then doing mod with binary 2 or 10.

Binary 2 looks like a 10 doesn't it? Yeah okay. So if I add 1 to 1 then I get 2, i.e. 10,

and then I take mod of 10 and that will leave me with 0. Same thing there.

0 and 1 added together is 1 and 1, that's 0. 0 and 0 added together is of course 0.

And that's 1 and that's 0 and that's 0. So you see another way of looking at this

is that I'm adding things together and then throwing away, just keeping the last

position. I only show you that because it's a nice realisation that what I did

here is really just a minuscule version of what I'm doing with the letters as

well. Here I have two possible representations and in an alphabet, Roman

uppercase, I have 26 possible representations. And these will become

our bases for doing calculations.

Now I'd like to get a little bit mathematical on you. We're going to

revisit the idea of key spaces and see how the mathematics works a little more.

Let's say that this is my plaintext. First of all we saw methods like

translating into Navajo that didn't have any keys at all. And there you can

understand that if our method is disclosed then we've lost everything.

But then we moved on to keys and our next idea that was the alphabet shift

method where of course we had 25, because we had to take away one, 25 different

possibilities of shifting our alphabet

with this tool. 25 different possible positions. So I can take that position,

there we go, and I can then start to encrypt this and then I'll get N and

I'll get E goes to K and so on. We've seen that before. The point is that you

should realise we have a key space of 25 here. Okay, later on I introduced the idea

of alphabet scrambling. We could take the alphabet that we mapped to and

then scramble all the letters so they're not in their usual order. And then we

made the point that is now 26 possible choices for the first position in the

mapped alphabet. And on the second position where we've already used up one

of those, that's 25 possible choices and after that we've only got 24 left and so

on and so on. So you recognise that as that. That's 26 bang we can say or 26

factorial. Factorial 26. Yeah, that's a very big number. It doesn't look like a

big difference but it is. Maybe I'll try and work it out and show you how big

it is. But we've got a lot larger key space. The other alternative that we've

discussed very briefly as I remember, that is, well, why take one shift for the

whole of my message? Another level of complexity could involve actually having

one shift there. Let's go like this, random, and we'll take that then. Where

are we? That's a shift of U and that ends me up at H with B. Good, I'll do

that. And then spin the wheel again and stop it and I'll get something like the

same thing. Just for argument's sake I'll spin it again. And E now goes to U, etc.

Okay, Z. There were two L's here. If I spin it again, hopefully, then I get another L.

And you see how two of the same letter will map to different things. And we

like that idea, don't we, because we're obliterating patterns. So, and so on. We

could, if we could do that, then all of a sudden we realise that we have 26

letters and each position gives us the choice from all of those 26 letters. Even

A would work here. I don't mind if I get an A here so that maps to O. It's still

obliterating the pattern. It doesn't give me any clues to that that was an

original O. So, so it is 26. How many letters do we have here? 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

letters. If I were to have a key that was that long and randomly pick all the

letters then that means I have a key space of that size. Just as you understand

the reasons why we're playing with these different methods, it is to ensure that

we understand the relationship to key spaces and that we can, when we want to,

find ways to grow the key spaces. Now that's the introduction. To start looking

at the idea first of the Visionnaire Cipher and then we'll move on to weird

things happening when we look at one-time pads.

Shift cipher maps every character into another character in one alphabet. Such a cipher is a monoalphabetic cipher. Preserves statistics of underlying message which a cryptanalyst can use to decipher message.

Vigenere cipher – multiple alphabets to generate ciphertext.

Key is a sequence of letters. The key letters applied to successive plaintext characters ,when end of key reached ,the key starts over.

Length of key – period.

Uses several key letters so it is Polyalphabetic cipher.

Weakness

Repetitions occur when characters of key appear over same characters of plaintext .

Index of coincidence measure difference in frequencies of letters in the ciphertext.

The Vigenère cipher, a method of encrypting alphabetic text through a simple form of polyalphabetic substitution, can indeed be broken, and one effective technique for doing this is the Kasiski Examination method. This method was introduced by Friedrich Kasiski in the 19th century, but it seems there might be a confusion regarding "BISHOP" as it's not a recognized term or name specifically associated with the Kasiski method. Let's go over the Kasiski method:

**Kasiski Examination Method**

1. **Identifying Repeated Sequences of Letters:**
   * The first step involves scanning the cipher text for sequences of characters that appear more than once. These repeated sequences suggest that they might represent the same plaintext segment encrypted with the same part of the key.
2. **Measuring Distances Between Repeated Sequences:**
   * Measure the distance between these repeated sequences. Specifically, count the number of characters between the repeated sequences. The distances are crucial as they suggest possible lengths of the encryption key.
3. **Finding Factors of Distances:**
   * For each measured distance, find its factors. The idea is that the length of the key is likely a factor common to several or all of these distances. For example, if distances are 10, 20, and 30, a common factor is 10, which might be the key length.
4. **Determining the Key Length:**
   * Analyze the factors to determine the most probable key length. The most common factor among these distances is likely the length of the key used in the Vigenère cipher.
5. **Breaking into Caesar Ciphers:**
   * Once the key length is known, the cipher text can be divided into as many groups as the length of the key, and each group is essentially a Caesar cipher. Each group represents a letter in the key.
6. **Frequency Analysis:**
   * Perform frequency analysis on each group. Since each group is a shift cipher, you can use frequency analysis to determine the shift and thus each letter of the key.
7. **Reconstructing the Key and Decoding:**
   * Once each part of the key is identified, you can easily decrypt the original text using the Vigenère decryption method.

**Advantages and Limitations:**

* **Advantages:** The Kasiski Examination is effective in breaking the Vigenère cipher, especially when there are repeated sequences in the text.
* **Limitations:** It might not work as well with very short texts, texts with no or few repeated sequences, or texts encrypted with a very long key.

This method marked a significant advancement in cryptanalysis as it was one of the first techniques to effectively break a polyalphabetic cipher.

One time pad

One time pad – It can in principle be unbreakable.

Variant of Vigenère cipher with key as long as plaintext chosen at random and doesn’t repeat.

**Key Requirements**:

* + **Length**: The key must be at least as long as the plaintext.
  + **Randomness**: The key must be truly random, not just pseudorandom.
  + **Uniqueness**: The key must be used only once (hence the name "one-time pad") and then discarded.
  + **Secrecy**: The key must be kept completely secret.

Weakness of one time pad – key must not be used more than once.

DES

Most important symmetric cryptosystems in history of cryptography.

Revisiting aspects of weak ciphers,Vigenere

The thing I'd like to show you now is a cipher, a very old cipher. We call it the Vigenère

Cipher, named after its inventor. It was invented in the 16th century, long before computers.

And it's based on the same kind of alphabet substitution that we've already discussed.

The first thing I'm going to do, however, is to point out that we early on discovered

that things like spaces, as we have here, are unwise because we've not yet in this method

taken care of spaces. Also, the fact that we can have uppercase and lowercase, that's

not very clever. Also, at the end you'll see we have a full stop there. These are things

that our method is so simple that we haven't even taken care of those. We could develop

the method to take care of those, but let's keep things simple and I'll simply have to

adapt. So my first adaption is to take away the full stop at the end. We can't have punctuation.

I'm running a little program that helps me out here. The next step is to make everything

uppercase and following that we find all the spaces and take those away. And as I do that,

I see I can still understand this text. It's not so bad. The message, the content isn't

spoiled at least. It's still there, it's just a little more difficult to cope with,

but I'm happy with that. Now the idea with a Vigenère cipher is that we have a key

that is comprised of letters. In this case, you'll see my key is on the next line and

it has six letters and I have to spell a word. You will understand this key as to be the

kind of mapping that I've spoken about in the previous video. To encrypt, I will add

the M to the T, add the A to the H, add the G to the E and do a mod 26 in case anything

gets to be bigger than Z. So that's the idea, but first you see my key is too short. So

my next step is to simply duplicate my key so it's long enough. And this is not only

long enough, it's a little bit too long. Okay, there are two things I could do about that.

I could either put some padding on the message. I could have three X's at the end or something

like that and try to remember that the X's aren't part of the message. And I say this

because it's actually the case that some real encryption methods will apply some padding.

It's not something that you should ever worry about unless you're digging deep into the

workings of these things. For the final user, you'd never notice. Having said that, I think

my solution here is easier. All I have to do to make the key match the plaintext is

to take a few letters away. Okay, so now we're set up. We're going to do the mapping

between the key and the cleartext to get our ciphertext. And that's a program that I can

start like this. It starts quite slowly so as you can see what's going on, but to be

honest it's just artificial delays that I put in there. So sooner or later it'll get

quick. Yeah, thank you. We're done. And there we go. We have on the one hand our plaintext,

our key and our ciphertext. And let's consider now what we know about key spaces. The key

space for this, if we know what the method is, if we understand, if a cracker were to

understand that I was using the Vigenère cipher and understand that I was using a key

of length six, then that is a key that is 26, sorry, a key space that is 26 to the power

six large. That's the sum of all my possible values of the key. Okay, but it doesn't quite

stop there because the Vigenère cipher allows you to have different lengths of keys.

So I just happen to choose six. It could have been one or two or seven or eight. And usually

you try to work with manageable sizes, but then you could understand that the possible

keys we have here are one, sorry, 26 to the power one plus 26 to the power two plus 26

to the power three plus 20, et cetera, et cetera, until you get tired of the length

of the key. So it really is a very big key space here. Certainly not one that you'd like

to attempt by brute force without a computer, as they didn't have in these days. Having

said that, I have to reveal that two guys in the 19th century in separate places, they

both found a solution, a method of breaking this. Now I've been going on and on about

key spaces, but if you think back to the things we've talked about, we also have to

make sure that our method is a good strong one, i.e. that it really does shake things

up, that it really does obliterate patterns. And the other thing we have to worry about

is that our keys should be entirely random. That is, we want to have a flat key space,

we want every single key to be just as likely. And a technique often used with Vigenère

was to use a name or a word or something as a key. And that's not so smart, because then

a lot of the keys will be more likely than others. So a cracker will be able to start

with the more likely keys before he tries the more unlikely keys.

Oh, by the way, one of the guys you've probably heard of who discovered the way to crack this,

his name was Charles Babbage. And if you don't know Charles Babbage, then shame on you, because

he's the first computer engineer, we could say. At least he designed the first mechanical

computer. And many of the ideas that he invented were actually put into the electrical computers

later on. And not only that, he was very good friends with the first visionary programmer

ever, Ada Lovelace, very famous figures. And it just so happens that Charles Babbage was

one of the guys who discovered the method for breaking this without having to go through

every single key in the key space. So you've guessed one of the faults with this method

already is because I've used the name Maggie. The other point I think you could probably

guess at the other fault if you simply stare at the ciphertext for long enough. So I'll

let you stare at that. And we'll see if we can discover where the other problem lies.

Cipher perfection – one time pad

Now, in class, I tried to give you an example of what I called a perfect cipher, one that

cannot be broken so long as it's done properly. And if you remember, I did that by writing

ones and zeros on the board and getting help with ones and zeros, and using XOR as encryption

algorithm. After that example, I think I saw a few blank faces. So in case it helps, I

thought I'd go through this again, but this time not using ones and zeros, but using letters.

We're getting used to using letters now. And I'm going to try and take it very deliberately

and slowly to see if I can bring to light why it is perfect, what problems it will create

for the cryptanalyst. So you'll see that my starting point is as with the Vigenère

cipher. The method is the same, pretty much. I also have a little program to help me. This

time I've called it OTP run. OTP is standing for one time pad. And I run this program.

And the first thing it does is it takes away the final full stop. Next stage, make it all

uppercase. Next stage, take away all the blanks. Thus. Okay. Now we don't have a key yet. So

let's start generating a key. The secret here is that it should be a random key. So this

is what I'm animating now, showing you how I'm trying to pick letters. I'm getting tired,

so we'll make it go faster. There we go. And that should be fairly close to a random key.

I see a worryingly large number of C's in there. So I start to doubt my random process.

But I'll ask you to at least imagine that's a very good random key. I've done my best,

I promise you. Right. Now that key is as long as the message. And that's an important point.

Remember in Vigenère, we had a short of key that we had to repeat. We had problems with that.

The thing I've done here will solve those problems. Next stage is simply to run the

encryption the same way as I did with Vigenère. And indeed, you should see none of the problems

I got with the Vigenère. There we go. Just to make things clear at this stage,

I'm going to make a note, call this plain text. This thing, remember, is the key.

And this is now our ciphertext. Very good. Do you believe me? Did that work?

Well, in case any of you doubt, I should run it backwards just to make sure that when decrypting,

I can get the same thing out again. So let's do that. I start with the ciphertext.

This is a symmetric algorithm. That means that we use the same key for both encryption and decryption.

And then I'll position myself correctly. Then I will run the decryption algorithm.

In this case, I've called it vigdec for Vigenère Decrypt. And we run that.

And lo and behold, no tricks. It really does decrypt to be the original message.

Good. So far. Right. Now, I have to convince you that there is nevertheless a problem.

And let's, to illustrate that, imagine down here that now I am the cryptanalyst

and I'm trying to do something with just the ciphertext. So this is a ciphertext attack.

I have no clue to the key. Yeah, I have no clue to the key.

So I've got to try and find the key. My best alternative is, of course, at this point, to do a brute force attack.

That means running through every possible single combination of the key.

Now, when you see our key here, you will realise that it's, well, if you count it, it's 81 places I've counted.

Trust me. So that means a key of that length actually has a key space of 26 to the power 81.

And that's a healthy key space, isn't it? Indeed, I start to doubt that even with the most powerful computers,

I will get anywhere with that. But nevertheless, I'm going to ask you to imagine that I do have the world's most powerful computer.

I have unlimited with time and I really want to decrypt this. How do I go about it?

Well, my first step then will be to generate the first key. I call this otpcrack.

That's not a very helpful program, to be honest. It's just filling out with the first key.

I start with that one and I go through all the possible combinations. And by the time I reach the end, I'll have all Z's there.

I do this in a methodical way to make sure I don't miss any of the keys. So I will try this key first.

You may already have guessed that as I do this, the result is not going to be very enlightening.

Nevertheless, let's give it a go. We are running vigdec. And as you might have guessed, a key like that has absolutely no effect.

But as I say, I'm being methodical. So I'm going to start with that just in case.

I see it has no effect because I can make no sense of what I see in front of me.

Let's try the next one. And that, in careful order, will be that key.

And while you've already guessed, most of that won't make any difference. It will make a difference for the very last letter.

But you get the idea. I'm not going to try it. But after I fail with that, I would go on to that.

And once I've been through the whole alphabet, I would go to that.

And if that still didn't work, then I step up one in the next place and go back to the beginning there and decrypt with that.

So you get the idea. It's going to take me a long, long time.

But I've got unlimited time. I've got an unlimited, powerful computer.

So what's the problem? Well, I shall try to illustrate the problem.

Now then, so far we've only looked at very trivial keys. I'm going to replace that with another one like this.

And I'll do as before - vigdec this.

That wasn't much good either, but you'll understand that this is one of the keys that I'll have later on.

And this will be my result. I'll have a careful look at that.

Is there anything I can... "BE" is a word. "Sylt". There's the word for jam in Swedish in there somewhere.

Otherwise, I can't see a lot that's making a lot of sense. And yeah, no, I think at this point, as cryptanalyst, I will say, no, that's mostly rubbish.

That's not the answer. And I keep on trying and I keep on trying.

But ultimately, I will reach other keys.

vigdec on this. Oh, this looks interesting.

Here we go. Roses are red, violets are blue.

This message of perpetual love is sent one time pad encrypted to you.

So it makes sense. This key applied to this ciphertext made this message, which is perfectly understandable.

So as a cryptanalyst, I might be jumping for joy now.

I think I found the answer. But you and I know that no, this up here was actually the original text.

So what's going on? I'll give you another example.

I'll just take away this. And I'll replace it with this.

I'll start up my decrypt again - vigdec.

And a completely different key. But as we see this, we see it's making sense again.

Danger, the rogue spy named Alan Davidson is set on world domination and shall destroy all you love.

OK. The point here is that given this ciphertext, since the key is as long as the ciphertext,

since the key can take on any single value, it will create every single possible combination of letters that are 81 letters long.

By the time it gets to the end of all Z's.

That means the problem is that you will never know when you found a text that seems to make sense, whether it was the original text or not.

You will find all sensible texts of 81 characters long as well as all the insensible ones.

So that's the reason why this can be regarded to be a perfect cipher.

So we have this perfect cipher. We can give it the name one time pad, which is one of its real names.

And ask the question, if this is perfect, why don't we use it?

This should be the be all and end all of ciphers. But for some reason we're using different ones.

So I'm still asking you to think about this. What's the problem with using this in real life?

I'll probably give you the answer in class.

A look at DES

Right then, we've got as far as looking at two very basic methods of encryption, that's substitution and transposition. And they, for a while, looked like little trivial toys, but then with a little twist of the imagination

I hope we can believe that both of these methods can become very strong indeed. In fact, when talking about the real methods that we're using today, you'll find that they are a mixture of these two ideas. They'll have a little bit of substitution and a little bit of mixing up and transposition. So, we've got that far to understand the nuts and bolts of what's going into the algorithms we're using.

We can understand how it's important to think about things like keyspace, and I hope we'll discover other things that are important to think about during class time. Right, so we can put all this together into an algorithm like, say, good

old DES. I'm going to discuss DES briefly. It's discussed in the course

book as well. But really we should be only looking at this as a

historically interesting method. DES isn't something that I would recommend

you to use today. There are much better algorithms and DES is, after all, getting

on for some 40 years old soon. Okay, but there's some interesting aspects of DES

that I'd like to introduce to you to learn more about algorithms in general.

A diagram of a flowchart

Description automatically generated

So, if we take a little, a slightly closer look at the DES algorithm. I'm not going to include an awful lot of detail, as you can see. At least in this picture we can see how we take our clear text. In this case it's marked as M0 M1 and we can imagine it carries on over here. I imagine that the M here stands for

message, but we're still talking about the clear text or the plain text.

Yep, we

put it through a little DES box here and on the other side we get blocks 0, 1, etc. of our ciphertext. So that should be clear. We're taking our key and applying

it to each of these little blocks. Inside this DES box or figure I've baked in all

the aspects of substitution and transposition. They're in there somewhere.

There's also things called S-boxes that are interesting to read about. If you are

interested in the inner workings of DES and even some of the politics behind

DES, then you can get better insight into DES by all means. But it won't be

part of the exam, so don't worry about it if you're not particularly interested.

Just understand there's the kind of mechanisms we've been discussing going

on in an advanced manner inside this box. Yeah, these blocks are incidentally 64

bits large. And on the other side we correspondingly get 64 bits large blocks coming out.

So that means we're chopping up our original plain text and then

sticking together our ciphertext at the end. Now this is an original picture of how DES originally worked. We call this electronic code book mode.

That's not especially important. ECB mode. Things we can understand from this

picture are for one. Well, what happens if we were to get the same 64 bits more than once by accident, if you like, occurring in the plain text? Well,

there's nothing that makes DES change its operation. It's exactly the same

operation each time. So if this were the same 64 bits as this 64 bits, that would

mean that the ciphertext here and the other end, that will have the same pattern.

You might not be able to understand this. You might not be able to understand that

easily. But what you can understand is that there is a pattern. By looking at

the ciphertext you can see that there is a pattern here that presumably comes

from the pattern in the original message. And that is a weakness because, as we

will hopefully all understand, patterns in our encryption methods are the

starting point for good cryptanalysts. If they can find something that is

repeated, that's a means for them to start doing statistical analysis of

what's going on and thereby try and work out, say, what your key was. On the other

hand, this does seem to work fairly well because in the right circumstances.

In Bishop's book you will read about stream ciphers and block ciphers. Well, this is, yes, it's a block cipher, but we could imagine that this is a streamed

message, a video message, and we can take 64 bits at a time and neatly translate

them, encipher them. It would work but it would be insecure in as much as we're not eradicating the patterns. The patterns that occur in the plaintext are

transferred as patterns to the ciphertext. Now then, we can make a clever

little twist and we can fix that problem. That's called CBC mode.

A diagram of a flowchart

Description automatically generated

And, well, it's the same picture that we have here, it's just that now we've introduced the idea of an initialisation vector. And this just happens to be 64 random bits, 64 1s and 0s that are generated at the beginning of the encryption along with the key. So we've got two elements going into the recipe. What happens is that if

I take my 64 bits from the plaintext and then mix it up, in what way we'll

probably have the chance to look at in class, but we mix those two together in some mathematical operation, and that's the input to the DES algorithm. We get out a ciphertext, but in the meantime we take a copy of the ciphertext and carryv it along with us. So in the next 64 bits we're going to mix in what we got here,

here, before it goes into the DES box. And then there's a little line here

indicating, yep, we continue to do the same thing. So the result from the second block encipherment will go on into the third and so on and so on. So that's a

neat little trick. You should understand that this improves the algorithm because it means if these two were the same we will have mixed things up so much so

that we won't get the same output.

Steganography

Do you recall when I spoke about the scytale? I mentioned that in ancient Greece, where

this method comes from reputedly, they didn't use strips of paper, they used a strip of

leather around a staff, wrote the message and wound the strip of leather. But that's

not exactly the whole story. In fact, the story goes that they would then take that

strip of leather and use it to tie their boots on with, up their leg or something like that, so that if anybody were to see the person who was transferring the message, taking the

message to its destination, then you wouldn't notice that there was a message there. Simple.

However, if we stop to think about it, this stage of the method, the earlier stage had

transposition in it, but now we're talking about this, wearing it on your body, that is clearly

a method of keeping a secret. Okay? But when I analyse it, it doesn't seem to have any elements

of substitution in it. No. Does it have any elements of transposition in it? No.

In that case, we already start to understand that this isn't what we've heard of so far.

This doesn't appear to be cryptography. This appears to be something else.

And indeed it is. It's something that's so different that we have another name for it.

And that name is

Steganography. And if cryptography is hidden writing, then we can say that this instead stands for

Covered writing. It's a good enough translation I've heard.

Now, in the early days of cryptography, they didn't actually make much of a difference between

these two ideas. There doesn't seem to be much of a difference between hiding something and covering

it. But of late, we've come to understand cryptography in fairly limited terms, the ones

we've already discussed. And so in the modern age, we make quite a clear difference between

cryptography and steganography. And the difference is that here, we know there's a message. What we're

doing with the message is we're hiding the content of it. I can see that there are lots of characters

there, but I don't know what the characters mean. Here, the idea is to actually disguise the fact

that there is a message at all. Therefore, the leather strips around the leg as a piece of

clothing. There are other classical examples of this, such as invisible ink is the classic one.

You can't even see that there's writing there until you treat it in some special way.

And you can probably think of other examples. A spy putting a plant pot in a window as a signal

to his accomplice to say, meet me at the town square at 12 o'clock tomorrow. An ordinary person

passing by the window would think, oh, there's a nice plant. Another person would see it as a piece

of information. So, classic steganography. <i>Now, I'm overdubbing this part where in the video,</i>

<i>I note that in the previous course book, the author does not address steganography.</i>

<i>But in fact, Bishop also does not go into it. Both authors go into the cryptanalysis technique</i>

<i>of traffic analysis that I tie into the discussion there. As Bishop says on page 383,</i>

<i>"A cryptanalyst can sometimes deduce information not from the content of the message,</i>

<i>but from the sender and recipient". I would add also that information can be gleaned from</i>

<i>attributes of a message, such as when it is sent, its length, etc. So, we have a kind of risk here</i>

<i>that is not solved by cryptography alone. </i> So, there you see, he identifies the risk.

And this is a risk where cryptography doesn't help us,

because the very fact the information exists can be the problem. An example could be,

imagine that you're analysing the data traffic from my accounts, and all of a sudden, you see

amongst all the encrypted information that some of it is going to just one of the students in the class.

And this is shortly before the exam. And then you notice by analysing the students' data

that there are transmissions going on to a Swiss bank account in connection

with this apparently. Right? You don't know what the content of any of this is, but the very fact that this

communication is occurring is perhaps even suspicious. It's telling you something.

Okay? So, this something else that you can divine by watching traffic,

we're going to give that a name. We call this traffic analysis.

This is the possibility that we can find out things just by the fact that there is traffic,

without even knowing what's in the traffic.

Hmm. So, having said that, it seems like there ought to be a place for steganography even in the

computing world, not just ancient Greece. And is there? Well, there is to some degree.

It's very difficult to implement in many ways. Possibly many of you have already heard about

steganography in terms of pictures. It has been said that terrorists have used the sending of

pictures, JPEG, electronic pictures, with hidden messages inside the code that creates the picture.

And you can play with tools that insert cryptographic, sorry, steganographic messages

into pictures. So, we've seen that kind of tool. Otherwise, what are the things you can do?

You could, well, you could imagine that in order to hide the fact there's a message between me and

another student, I might have to constantly send rubbish information to that student or to other

students to hide the fact that once in a while a transmission is actually a real transmission with

real content. A lot of work, as I say. Nevertheless, it is a concept that is useful to know. It is a

concept that has occurred on exams before. So, that's why I couldn't let you pass without meeting it now.

Separate channels

I've become aware that in the videos I've started to mix up the idea of

plaintexts and messages, as we saw in the desk slide. I'd just like to make one

point clear before we move on then. Say that we've got one party here who is

encrypting things. Well, should I call it a plaintext or should I call it a message?

What's in a name? It's not that important what we call it so long as we understand

what it is we're doing with it. The reason message comes into the picture a

lot is because, of course, a very normal case is that A is sending a secret to

somebody else and therefore it seems like it's very often a message. But

nevertheless I want to make the point that sometimes what we'll be doing is

encrypting things for our own sakes. I did this right at the beginning. I made a

secret note in my own diary so that nobody could eavesdrop the information

in my diary, but it was for me. So I just want to make that clear that when we

start talking about messages we maybe shouldn't always assume that

cryptography is about messages. Sometimes it's about encrypting things for

yourself. We often encrypt our hard drives. Those aren't messages, those are

plaintexts. Okay, having said that, from now on I think I might use plaintext and

message interchangeably, but just as long as you don't assume that

cryptography is always about sending messages, a lot of the remaining will be

about sending messages. So if you've been reading the book carefully you will

have already guessed that all the things I've been talking about in the video

films, these encryptions I've been doing, they've been symmetrical, haven't they?

What that means is that I am taking, I'm going to call it a message now, say I'm

taking a message and I'm doing something with that. I am encrypting it.

Now I am encrypting it with a certain key. I send the message to the receiving

party and the receiving party, oh sorry, oops, don't go too far, too fast, too far.

This will give us, let's make it an arrow, it could be equals, but that will give us

what we call our ciphertext. Yep, so it's the ciphertext that we, getting mixed up,

there we go, it's the ciphertext that we're sending to C, of course, so C takes

the ciphertext, does something with that, and what we call that is decrypt the

symmetrical parties. In order to decrypt we need the same key. This is why we call

it symmetrical cryptosystems because it's the same thing on both sides, the

same key. That point will become clearer when I show you the alternative. Right, so

aren't we happy with symmetrical cryptosystems? They seem to work well,

they seem to spoil patterns, they seem to give us nice large key spaces. Maybe

that's what we need. In practice I think you'll find that we will have problems,

primarily with the idea that if I'm now sending something here, I've got to fill

this in so that B can decrypt it and get the message back, then I send C over

here, but in some way I also have to make sure that the key I've generated to do

this has to move over there as well. A little hint in case you haven't already

guessed it, that's the problem. That's the main problem with the idea of the

one-time pad, moving the key. Anyway, so therefore we have to have these things

like key exchange protocols to move the keys, and that adds an extra degree

of complexity to our whole picture is how are we going to get the key over? A

common way to do things safely, securely, is to use a second channel. Let's say I

send the crypto text over email, whatever, yeah, I send the crypto text over email.

In that case I probably wouldn't be, it wouldn't be very wise for me to send

this key in the same email package, or it wouldn't be wise because anybody who

eavesdrops will get a hold of the key and now secret is lost. Okay, so a thing

you'll often find is that when being extra secure I can send one thing in one

channel and the other thing in the other channel, and the person at the other end

has to match the two data sources. So I could maybe phone you up and read to you

what the key is over the phone. Now somebody could be eavesdropping the

phone as well, but the likelihood that they both be eavesdropping my email and

my phone are less than if they were eavesdropping one channel. So I can play

that game, use separate channels to do things. Do you notice the similarity to

the idea of defense in depth? Anyway, that's one thing we can do. We can

use a different channel, whatever it is. I could send it by old post in a letter,

write it down, things like that. Separate channels. When I'm talking about reading

keys over telephones and sending you on paper, then it starts to sound like this

movement of keys really is a pain and we ought to try and find a better way to go

about it. There are better ways to go about it. You can read about key exchange

protocols in the book. There's another problem with symmetrical

curricular systems that I'd like to discuss. We've seen how it's a problem

with moving the keys when you're dealing with messages. But not only that.

Let's say that party A needs to speak to B. As we've seen in that case, A somehow

needs to create a key and move it over and share that key with B to share

secrets. But consider that there aren't just two parties in this world. There

should be several in a normal course of events. And of course if there are

several people and I want to share people, share secrets with them, then I'm

going to have to create another key for that person and another key for this

person. And I've already run out of different colours and I'll need an awful

lot of colours to keep them separate. And not only that, these people will be

wanting to communicate with each other. So we've got B communicating with C and

D communicating with B. And we've really got this situation of people wanting to

communicate with each other. So this is my set of keys shared with those and

this person will have their set of keys shared with those and this with those.

And if I want to have secrets in groups of people, then I'm very quickly going

to have an explosion of different keys and have the problem of keeping track of

whose keys belongs to who and things like that. So the maths of symmetrical

cryptosystems looks neat but we're still seeing the problem of keys. So it would

still be nice if we didn't have to deal with them by reading them up over the

phone and things like that, wouldn't it?