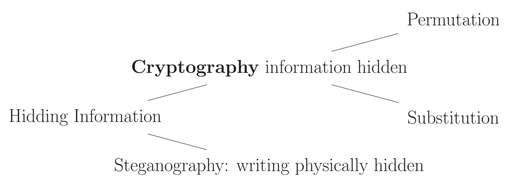
Ciphers and Attacks

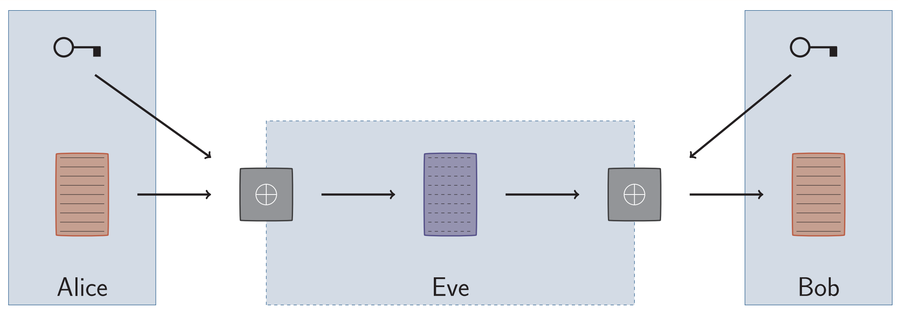
Arne Hansen and Prof Stefan Wolf
Cryptography and Quantum Information, USI Lugano

Categories of Secrecy



Cryptography hiding content of a message without hiding the writing itself by replacing letters with others (substitution) or scrambling the letters of the message (permutation).
Steganography physically hiding the message (invisible ink, ...)

The Scheme of Cryptography



First a player, usually called Alice, encrypts the plaintext according to the encryption algorithm using her key. The ciphertext Alice generated is then sent to another party, often called Bob. With his key he can execute the decryption algorithm in order to obtain the plaintext again.
Security What does it mean for a cipher to be secure? The notion of security depends on the abalities we imagine an adversary has to attack the cipher. The adversary, often referred to as Eve, can at least read the ciphertext, i.e. tap the wire between Alice and Bob. Usually Eve also knows the encryption and decryption algorithms. Obviously Alice and Bob have to keep their key secret. If Eve's knowledge of the algorithms and the ciphertext suffices to retrieve the plaintext, the cipher is insecure. Otherwise Alice and Bob can communicate secretely.
Authenticity What if Eve doesn't merely read the ciphertext but also changes it? Or sends something to Bob, claiming to be Alice? These considerations lead for instance to *digital signatures* aiming to insure Bob that what he had sent was not altered or faked.

Scytale: a transposition cipher

The message "Help me, I am under attack" is written on the scytale in rows

H	E	L	P	M
E	I	A	M	U
N	D	E	R	A
T	T	A	C	K

After unwinding the band it is scrambled to

`HENTEIDTLAEAPMRCMUAK`

The scytale is thus a permutation cipher.

The Caesar Shift Cipher

The simplest substitution cipher is the Caesar Cipher. Caesar shifted all the letters down the alphabet by a fixed step, say for instance 3 letters. The plaintext and ciphertext alphabet are associated as follows

Plaintext: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
Cipher: `XYZABCDEFGHIJKLMNOPQRSTUVW`

An example encryption would be

```
Some important message.
PLJB FJMLOQXKQ JBPPXDB.
```

Vigenere Cipher

In order to reduce redundancy and thus prevent frequency attacks several substitution schemes were used one after the other. For instance the first letter of the plain text would be encrypted with Caesar shift of 3, the second with a shift of 12, ... and eventually starting again with a shift of 3. Once the number of different substitution schemes is known, the cipher is again vulnerable to frequency attacks. Taking a computer or a cipher machine, like Enigma, one can increase the number substitution schemes immensely and thus security.

Frequency Analysis

The letters of the alphabet do not occur with the same frequency. In English the letter "e" is the most common and occurs a lot more often than for instance "z". So if one knows in what language a plaintext was written, one can decrypt the ciphertext from a monoalphabetic substitution cipher without knowing the substitution scheme, that is the key. One merely has to count the number of occurences of any letter of the alphabet, compare this with the frequencies of letters of the given language and match corresponding letters. This works better the longer the cipher text is. (Actually if the ciphertext is very short, the cipher might become a one-time pad which is really secure).

Side Channel Attacks

While frequency attacks or brute force approaches (i.e. trying out a large number of keys) aim at the cipher itself, side channel attacks exploit weaknesses in the implementation of the cipher. For instance one might surveil the power consumption of the processor during encryption. This might reveal patterns that allow to break the cipher.

The One-Time Pad: Perfect security

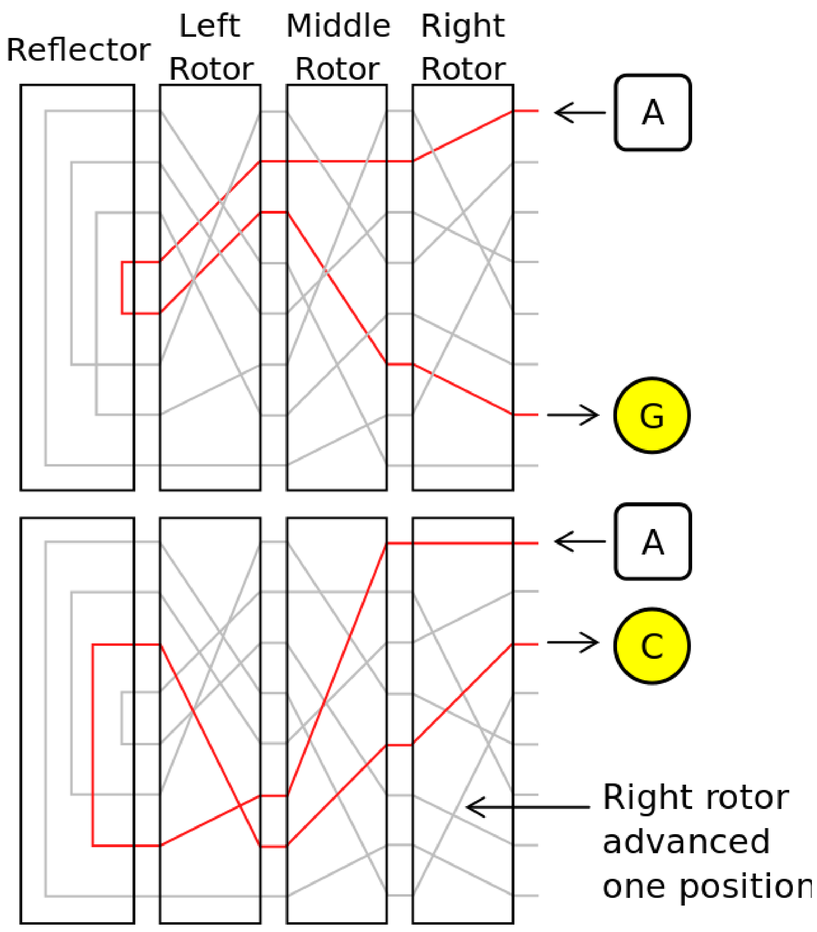
How could we get a substitution cipher with proper security? The issue of most substitution ciphers is their redundancy. For example the letter "e" might always be substituted by "m". This clearly allows frequency attacks as the number of occurences of "e" is just shifted to "m". In order to get rid of any redundancy or other structures that might be helpful to break the cipher the one-time pad uses an entirely random key that is just as long as the message itself. A binary example as it occurs in any computer is then

Text	`Ciao`
Plaintext	`01000011 01101001 01100001 01101111`
	$\oplus$
Key	`10011011 11110011 10011100 11011011`
	gives
Ciphertext	`11011000 10011010 11111101 10110100`

So adding the random key (i.e. applying the XOR (either ... or ...) operation) makes the ciphertext look completely random too. Eve — not knowing the key — could not say whether the ciphertext would stem from the word 'oggi' or the word 'ciao' (or any other four letter word), as they are all equally likely to appear. Thus the one-time pad is properly secure.
But — as the name already suggest: never ever use it even twice. This would introduce structure and therefore turn the cipher insecure.
Why isn't the OTP used all over the place if it is completely secure? Well, you need to distribute a lot of keys secretely. This is very inconvenient.

Enigma: a first machine cipher

The Enigma was the first electronic device used for encryption enabling far stronger ciphers. The idea was similar to the Vigener Cipher to continuously change the substituion scheme. As the number of possible schemes was rather large this encreased security considerably (provided the implementation did not reveal any information). In a sense the enigma could be considered as a predecessor of modern computers.



The core of the Enigma contained three disks with internal wirings. Every time a key on the keyboard was hit the first rotated by 1/26 of a circle. After 26 rotations of the first disk the second rotated once and after 26×26 keystrokes the third one moved forward. If a key was hit, a small current basically followed the internal wirings until a lamp indicated the corresponding letter of the cipher alphabet. The cipher alphabet changes with every keystroke and repeated only after 26^3 input letters.
The "mirror" on the left enabled to use the machine as its own decipher machine. Therefore the key contained the simply the initial settings, i.e. which disks were to be inserted at what position and with which initial rotation.

Blockciphers: Lucifer

All previous ciphers processed the plaintext letter by letter (and are thus called **stream ciphers**). Current ciphers encrypt entire code blocks, that is a fixed number of 0's and 1's in a computer, in each encryption step. Still the idea remains the same. A code block of 64 bits is mapped to another code block of the same size according to a scheme that is dependent on some key. The number of possible schemes has to be sufficiently large and each scheme has to be equally like to occur, in order to insure security.

Key Distribution

All protocols mentioned so far require that Alice and Bob share a secret key only known to them. This usually meant Alice and Bob had to meet at some point and actually agree on that key. As it turned out they could as well use a public channel to agree on a key. This seems a little paradox but imagine: Alice sends Bob a box secured with padlock with just one key. Bob adds another padlock with just one key and sends the box back. As soon as Alice is sure that the other padlock is Bob's she removes hers and sends the box agai to Bob who can now open the box and access the key. Something similar can be done with certain mathematical constructs.
Alice and Bob select each random numbers which they keep to themselves. But they share the result of a computation using those numbers publicly. Both can now use the value of the other to compute a third value using again their intial numbers. The both end up with the same value while someone else can hardly guess it from the public known numbers.

Asymmetric Encryption

In order to overcome the key distribution issue one can as well introduce a cipher system that relies on two keys — a public and a private one. If Alice wants to send a message to Bob she looks up his public key in some sort of directory. She encrypts her message using this public key. Now Bob is the only one to decrypt the message again using his private key.

Quantum Key Distribution

Very small objects like electrons (radius $3 \times 10^{-15} m$) or photons do not behave like a football anymore but rather according to the principles of quantum mechanics. Sometimes they rather behave like waves that can sort of cancel out one another and then again they have very football-like features like momentum.
One of the characteristics of quantum mechanics is that a measurement induces a change of the object to be measured. Thus measurements leave traces and can be detected afterwards in a suitable setup. So imagine Alice could encode some information like a cipher key in a quantum state and send it to Bob. Bob performs some measurements. By comparing what Alice has setup with what Bob has measured they can determine whether someone has taken a look at the system or not. Instead of sharing all their measurement results over a classical public information channel like an insecure internet connection they could share only a part of the measurements. If noone spied their system they can take the other part of the measurements as a shared key for e.g. the one-time pad to be really secure.
How is a measurement result, which could be just one number, enough for a shared key? If it was just one big number Alice and Bob would simply transfer it into the binary system, i.e. compute the representation of the number in 0's and 1's. Then they can use it for the one-time pad as before.

Quantum Computers

Not only cryptographers but also cryptanalysts would profit from a quantum revolution in computing. Modern ciphers are build on the assumptions that certain mathematical operations like the so-called factoring take a long time to be calculated on a computer. While this might hold for classical computers there exist already efficient algorithms for quantum computers compromising current ciphers once operational quantum computers are built.

Encryption Tools

The GNU Privacy Guard provides an open source implementation of OpenPGP. There are a number of frontends for e.g. email or chat clients.