Ca' Foscari University
Faculty of Computer Science

Cryptography

Afternotes

Author
Francesco Vivian

A.Y. 2020-2021

Lecture 1

Definition 1. Security: it generically refers to the possibility of "protecting" information, which is either stored in a computer system or transmitted on a network.

Whatever will be shown works in both situations.
To decide whether a computer system is "secure", you must first decide what secure means to you, then identify the threats you care about. Some threats are: cyberterrorism, denial of service, modified databases, virus, identity theft, stolen customer data, equipment theft, espionage.

There are different aspects to protect through security properties:

Property 1. Authenticity: an entity should be correctly identified.

Example 1. Some examples of authenticity:

- login process for authenticating a user (User Identification)
- a digital signature allows for authenticating the entity originating a message (Message Authentication)

Property 2. Confidentiality (secrecy): information should only be accessed (read) by authorized entities.

- Confidential information is not disclosed to unauthorized individuals (Data confidentiality)
- Individuals control what information related to them may be collected and stored and by whom that information may be disclosed (Privacy)

Example 2. Some examples of confidentiality:

- the person that accesses a database should be authorized to access the data
- personal privacy, my private data should be protected while browsing the web

The "access control" for confidentiality:

- use the "need to know" basis for data access. How do we know who needs what data? (Approach: access control specifies who can access what). How do we know a user is the person she claims to be? We need her identity and we need to verify this identity (Approach: identification and authentication).

- Analogously, the "need to access/use" is the basis for access to physical assets (access to a computer room, use of a desktop)

Confidentiality is difficult to ensure and easy to assess in terms of success: it is binary in nature (Yes/No).

Property 3. Integrity: information should only be modified by authorized entities.

- information and programs are changed only in a specified and authorized manner (Data integrity)
- a system performs its intended function, free from unauthorized manipulation (System integrity)

Example 3. We should not alter bank accounts and IoT device firmware.

If we don't have integrity, we also don't have confidentiality (with integrity I want only authorized users to be able to modify information, with confidentiality I want only authorized users to be able to see information, modifying includes seeing). Integrity is more difficult to to measure than confidentiality, it is non binary (it has degrees of integrity) and it is content-dependent (it means different things in different context)

Example 4. A quotation from a politician, we can preserve the quotation (data integrity) but mis-attribute (origin integrity), like *Y said that* instead of *X said that*.

Property 4. Availability: information should be available/usable fastly by authorized users.

Example 5. It is important to guarantee reliability and safety. Apart from attacks, availability might be loss in case of faults (we need to use fault-tolerant techniques). We need availability in case of a remote surgery for example (good QoS).

We can say that an asset (a resource) is available if:

- it provides a timely request response
- it provides fair allocation of resources (no starvation)
- it is fault tolerant (no total breakdown)
- it is easy to use in the intended way
- it provides controlled concurrency (concurrency control, deadlock control, ..)

Property 5. Non-repudiation: an entity should not be able to deny an event.

Example 6. Having sent/received a message. This property is crucial for e-commerce, where "contracts" should not be denied by parties.

Other properties that are not addressed in detail are: fairness of contract signing, privacy, anonimity and unlinkability, accountability, ..

Typical attacks

We will now see some typical attacks. We assume that information is flowing from a source to a destination (e.g.: reading data is a flow from the data container to a user, writing is a flow from a user to the file system).

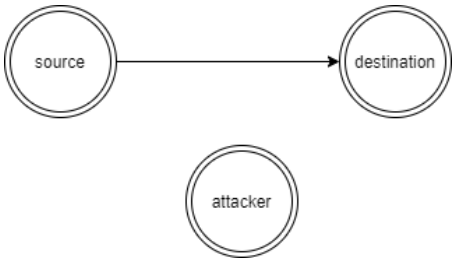

Figure 1: Expected information flow

Malicious users might try to subvert the properties previously mentioned in many different ways. We will now give a general classification depending on how an attacker might interfere on the expected flow of information (Figure 1).

Definition 2. Interruption: the attacker stops the flow of information (Figure 2). The attacker interrupts a service, it breaks system integrity and availability.

Some examples of interruption:

Example 7.

- the destruction of a part of the hardware
- canceling of programs or data files
- the destruction of a network link
- a denial of service (DoS) that makes the system/network unusable

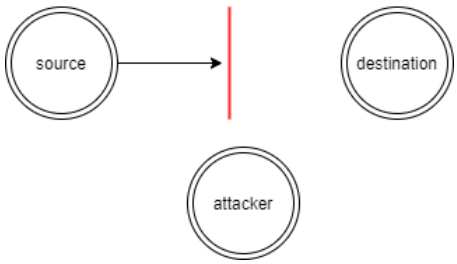

Figure 2: The attacker interrupts the flow of information

Definition 3. Eavesdropping (interception): the attacker gets unauthorized access to the information (depicted as an additional flow towards the attacker).

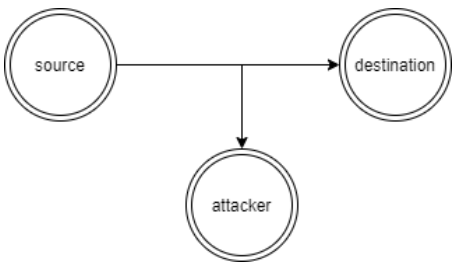

Figure 3: The attacker intercepts information

Interception is an attack to confidentiality, these attacks are hard to detect.

Example 8.

- unauthorized copies of files or programs;
- interception of data flowing in the network (a credit card number).

Interception attacks are hard to detect because source and destination don't notice any change (differently from interruption, where destination don't receive the flow).

Definition 4. Modification: the attacker intercepts the information and make unauthorized modification.

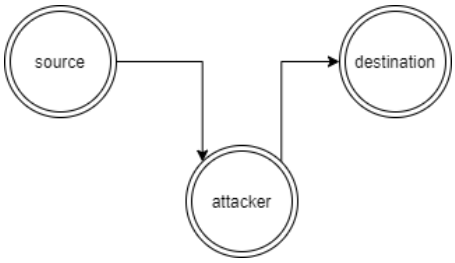

Figure 4: The attacker modifies information

In this case destination might notice that something is wrong.

Example 9.

- unauthorized change of values (e.g.: of a database);
- unauthorized change of a program;
- unauthorized change of data flowing in a network;
- A redirects S's bank transfer to herself (either in the browser or in the network, man in the middle).

Definition 5. Forging (falsification): the attacker inserts new information in the system (usually related to impersonation since the attacker lets the destination believe the information is coming from the honest source).

Forging is an attack to *authenticity, accountability* and *integrity*.

Example 10.

- addition of messages in the network;
- addition of a record in the database.

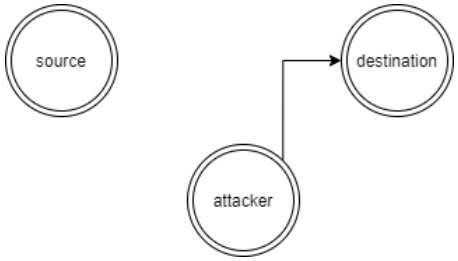

Figure 5: The attacker forges new information

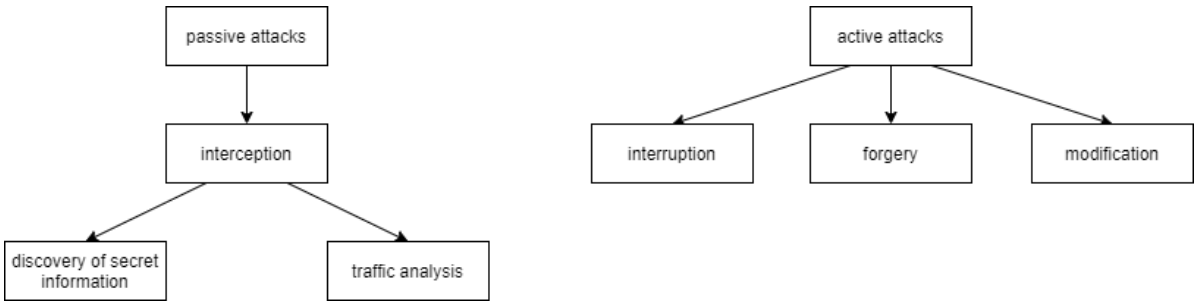

Figure 6: Classification of different types of attacks

Example 11. Suppose a bank B is using the following simple protocol to allow a bank transfer from user Alice (A):

$$\texttt{A} \rightarrow \texttt{B: sign\_A("please pay Bob 1000€")}$$

`sign_A` is some "signature" mechanism to ensure that the message really comes from Alice (thus the attacker cannot generate valid signed messages from Alice).

Let's suppose Bob is the attacker, he intercepts the whole message and repeats it as many times as he want, without modifying it. This attack (called replay) consists of an interception plus forging (in this case the message is just re-sent as it is). Bob obtains many bank transfers by just re-sending message M.

Example 12. Program modification: It is an attack to confidentiality, Bob modifies a program that is used by Alice, such a program apparently works normally, however it changes (e.g. the access rules of the users that are executing it, in this case it is called *Trojan horse*). Bob waits that Alice uses the program and copies all the files of Alice in his home directory.

Cryptography

The term *cryptography* comes from the greek and means "hidden writing". It is a way to protect the information when the environment is insecure. For example it is used when the information is sent on a network such as internet or when the system does not support sufficient protection mechanisms.

Definition 6. Encryption: a *plaintext* (message) is transformed using some rules (encryption algorithm) in a ciphertext.

Definition 7. Decryption: the plaintext is reconstructed starting from a ciphertext.

The decryption has to be simple for the receiver and unfeasible for an attacker. The information is encrypted in the source and travels to the destination where it will be decrypted. In order to do this there are two possible solutions:

1. only the sender (Alice) and the receiver (Bob) know the encryption algorithm. If the attacker, by looking at the flow of information, is able to guess which algorithm is used, then he will be able to decrypt all the messages sent;
2. the encryption algorithm is public and Alice and Bob share some information (the encryption key) non accessible by the attacker. If the attacker doesn't have the encryption key, it is unfeasible to decrypt the messages.

The second solution is better because it is simpler to distribute only one key and if there is an attack it is easier to change key instead of a whole algorithm.

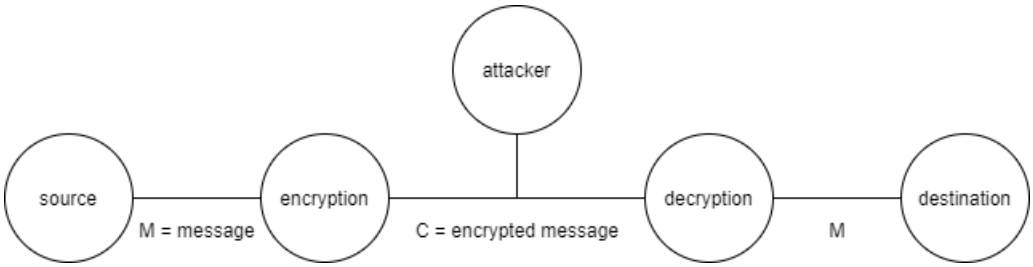

Figure 7: Encryption with a shared key

We want to build a secure channel to be able to exchange the encryption key.

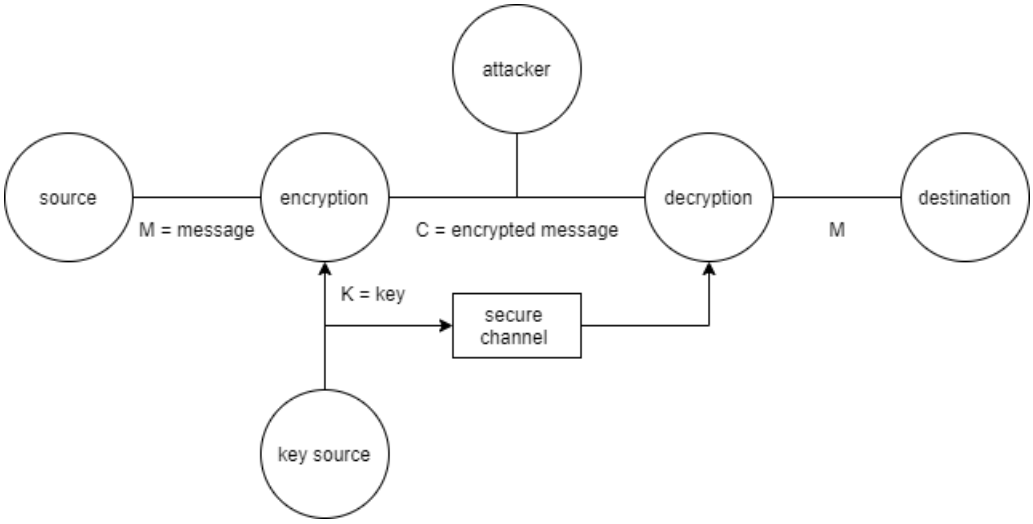

Figure 8: Encryption with a shared key and the secure channel

This is called *symmetric key cipher* (symmetric because source and destination use the same key).

One of the first encryption algorithms was used by Julius Caesar. In the Caesar Cipher all the letters are permuted using a certain rule (each letter is substituted by the one 3 positions ahead in the alphabet)

$$\begin{aligned} \mathtt{A} &\rightarrow \mathtt{D} \\ \mathtt{B} &\rightarrow \mathtt{E} \\ \mathtt{C} &\rightarrow \mathtt{F} \\ &\ldots \\ \mathtt{Z} &\rightarrow \mathtt{C} \end{aligned}$$

In this case the algorithm is the Caesar Cipher and the key is 3.

Lecture 2

The idea behind this course is to build step-by-step a system that is stronger and stronger until we'will arrive to systems that are used in practice.

Defining a cipher means to define an encryption algorithm and a decryption algorithm. A **cryptosystem** (or **cipher**) can be defined as a quintuple `(P,C,K,E,D)` where:

- `P` is the set of plaintexts (i.e. all the Italian words);
- `C` is the set of ciphertexts;
- `K` is the set of keys (we can have more than one key);
- `E:K×P→C` is the encryption function;
- `D:K×C→P` is the decryption function.

Let `x∈P, y∈C, k∈K`, we will write E_k(x) and D_k(y) to denote E(k,x) and D(k,y), the encryption and the decryption under the key k of x and y respectively.

We require two properties for each cipher that uses a shared key:

Property 6. $D_k(E_k(x)) = x$, decrypting a ciphertext with the right key gives the original plaintext.

Property 7. computing k or x given a ciphertext is infeasible, so complex that cannot be done in a reasonable time.

All the ciphers we will discuss have the first property, only "secure" ciphers have the second one (Caesar cipher doesn't). If someone, one day, will prove that P=NP, then most of the ciphers won't be secure anymore.

Example 13. Referring to the Caesar cipher, we can define the encryption function as "the letter three positions ahead (of our letter x) in the alphabet" or `(x+3) mod26` and the decryption function as "the letter three position behind (of the letter to be decrypted)" or `(x-3) mod26`. Our key k=3.

If we find the message `"BHV BRX PDGH LW"` and we know that it is encrypted with the Caesar cipher we can easily decrypt it into `"YES YOU MADE IT"` just going back 3 positions. We can notice that we have two "B" and they correspond to the same decrypted letter "Y", in monoalphabetic ciphers this is a strong weakness.

Proof of the first property applied to the Caesar cipher. We have to prove that:

$$\begin{aligned} x &= D_k(E_k(x)) && (1)\\ &= ((x+3) \bmod 26 - 3) \bmod 26 && (2)\\ &= ((x+3) - 3) \bmod 26 && (3)\\ &= x \bmod 26 && (4)\\ &= x && (5) \end{aligned}$$

Since we have the modules repeated twice we can keep only the external one. ■

To prove this property we can apply this reasoning to every cipher.

Definition 8. Kerckhoffs' principle: a cipher should remain secure even if the algorithm becomes public.

Kerckhoffs rules (1883):

- The system should be, if not theoretically unbreakable, unbreakable in practice;
- The design of a system should not require secrecy, and compromise of the system should not inconvenience the correspondents (Kerckhoffs' principle);
- the key should be memorable without notes and should be easily changeable;
- the cryptograms should be transmittable by telegraph;
- the apparatus or documents should be portable and operable by a single person;
- the system should be easy, neither requiring knowledge of a long list of rules nor involving mental strain.

The Caesar cipher is clearly insecure (it will be proved later) since once the cipher has been broken any previous exchanged message is also broken (as the cipher works the same way), the key should be changed and it is assumed to be the only secret.

Shift cipher

We can extend the Caesar cipher to a shift cipher with a generic key k, we can choose any key in the range $0 \leq k \leq 25$. For simplicity we will consider letters as numbers (`A=0, B=1, .., Z=25`), this means that $P = C = K = Z_{26}$ (Z_{26} is for all the integers between 0 and 25). For the encryption and decryption function we have:

$$\begin{aligned} \mathrm{E}_k(x) &= (x+k) \bmod 26 \\ \mathrm{D}_k(y) &= (y-k) \bmod 26 \end{aligned}$$

The Caesar cipher is a subcase of a shift cipher with k=3. In a shift cipher is useless to have k=0 because we would end up with equals plaintexts and ciphertexts.

Example 14. Considering k=10, it gives the following substitution:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
          ↓
KLMNOPQRSTUVWXYZABCDEFGHIJ
```

Proof of the first property applied to a shift cipher. We have to prove that:

$$\begin{aligned} x &= D_k(E_k(x)) && (1)\\ &= D_k((x+k) \bmod 26) && (2)\\ &= ((x+k) \bmod 26 - k) \bmod 26 && (3)\\ &= ((x+k) - k) \bmod 26 && (4)\\ &= x \bmod 26 && (5)\\ &= x && (6) \end{aligned}$$

■

Note that Z_{26} is a group under the addition (but not under the multiplication).

Definition 9. A **group** $\langle G, * \rangle$ is a set G together with a (closed) binary operation * on G such that:

- the operator is associative $((x * y) * z = x * (y * z)$ for all x,y,z in $\langle G, * \rangle)$;
- there is an element e∈G such that $a * e = e * a = a$ for all a∈G. Such an element is the identity element;
- for every a∈G, there is an element b∈G such that $a * b = e$. This b is said to be the inverse of a with respect to *. The inverse of a is sometimes denoted as a^{-1}.

The set $\langle Z, + \rangle$, which is the set of integers under addition, forms a group:

- addition is associative $((x + y) + z = x + (y + z)$ for all x,y,z in $\langle Z, + \rangle)$;
- the identity element is 0, since $0 + a = a + 0 = a$ for any a∈Z;
- the inverse of any a∈Z is -a.

A group that is commutative with an additive operator is said to be an abelian group.

The set $\langle Z, \cdot \rangle$, the set of all integers under multiplication, does not form a group. There is a multiplicative identity 1 but there is no multiplicative inverse for every element in Z.

From now on we will work with abelian groups.

Possible attack to shift ciphers

If I can attack a shift cipher, I can implicitly attack the Caesar cipher.

Example 15. If I am an attacker, I see the message `NGPPS` and I know that it is encrypted using a shift cipher but I don't know which key k is used I can try to get it by trial and error. I start with `k=1` and if decrypting the message I obtain something nonsense, I try with k=2 and so on. with k=4 I will reach that `NGPPS=HELLO`. Is it feasible? Yes because we only have 26 possible to keys to try. This type of attack is called **Brute force**.

In this case the problem with our encryption algorithm is the very small number of keys. Thus the second property doesn't hold (Kerckhoffs' principle: a cipher should remain secure even if the algorithm becomes public). The weakness is that we know that each letter is moved by the same distance.

Substitution cipher

A substitution cipher is a generalization to overcome the previous limitation: instead of moving all the letters by the same distance, now we use a generic permutation of the alphabet to map the letters. For example:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
          ↓
SWNAMLXCVJBUYKPDOQERIFHGZT
```

"HOME" becomes "CPYM". In this case the key is the complete permutation, otherwise the receiver is not able to decrypt our message. As for the previous ciphers, to decrypt we just apply the inverse substitution.
We have P=C=Z_{26} and K={p|p is a permutation of 0,..,25} with:

- E_k(x)=p(x);
- D_k(y)=p^{-1}(y).

Can we "Brute force" also this type of ciphers? No, we would have to try 26! keys, which is approximately $4 \times 10^{26} > 2^{88}$. This number of keys is very heavy to brute force, even with powerful parallel computers. We have to find something different to attack substitution ciphers. How can we do it?

It is a monoalphabetic cipher (it maps a letter to the very same letter), this preserves the statistics of the plaintext and makes it possible to reconstruct the key by observing the statistics in the ciphertext. For example, in Italian, almost all the words end with a vowel, and the vowels (a,e,i,o,u) are easy to identify as they are much more frequent than the other letters.

Let us assume a cryptoanalyst knows the used cipher (e.g. a monoalphabetic substitution cipher) and the language used in plaintext (e.g Italian), but he doesn't know the plaintext and the key.

Example 16. Given a ciphertext C, let us compute the frequency of letters, for example lette S appears 0 times and letter C appears 15 times. Is it possible that the cipher transforms A into S, and Z into C (A is never found and Z is found 15 times)? It is possible but very unlikely.

To compute the statistics for letters in Italian language it has been used a text of 14.998 letters, 1/3 from the book "Pinocchio" and 2/3 from the book "Il nome della rosa". The letters frequency are reported in the following table.

Letter	Absolute frequency	Percentage frequency
A	1714	0.114
B	160	0.011
C	637	0.042
D	566	0.038
E	1658	0.111
F	141	0.009
G	272	0.018
H	160	0.011
I	1563	0.104
L	966	0.064
M	436	0.029
N	966	0.064
O	1486	0.099
P	421	0.028
Q	85	0.006
R	978	0.065
S	771	0.051
T	1024	0.068
U	528	0.035
V	343	0.023
Z	123	0.008
Total	**14998**	**0.998**

Table 1: Letters statistics for Italian language.

The most used letter is A and the less used is Q. We can also build the graph in figure 9 to represent this statistic.

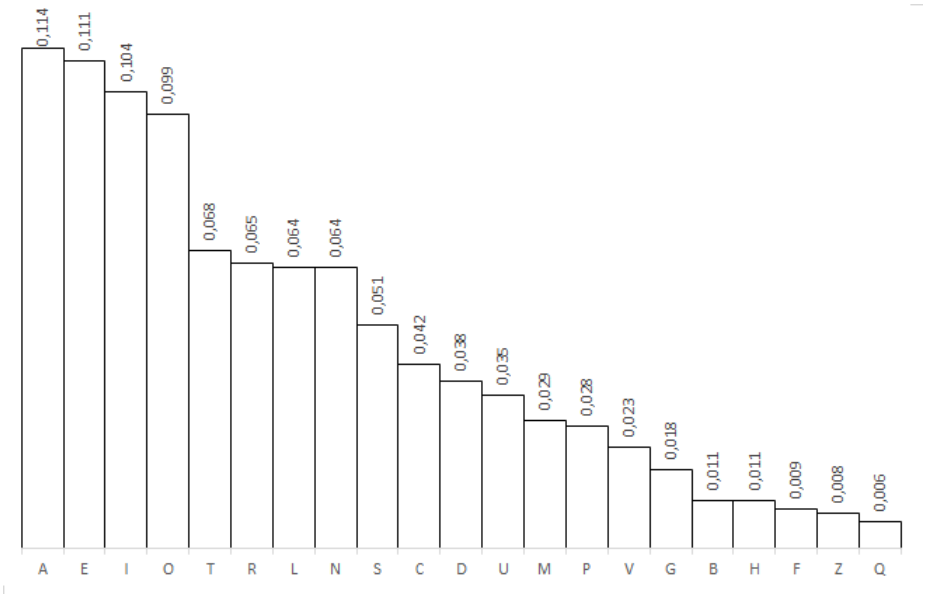

Figure 9: Percentage frequencies of letters in Italian language.

Knowing this, we can study the frequency of each letter in my ciphertext and map them using the frequency of my language.

- In Italian the frequency of letters I and L increases at the beginning of the sentences (articles "il", "lo", "la"), and the frequency of "A", "E", "I", "O" at the end of the words;
- there are words that may appear more frequently (e.g. the word "airplane" in a military message);
- there exist digraph and trigraph that are more frequent.

How can we decrypt a message?

- we order the letters of the ciphertext into decreasing frequencies;
- we substitute with letters in decreasing order as in the corresponding tables (depending on the language);
- there might be mistakes with letters that have the same frequency ("N" and "L" or "H" and "B" in Italian).

We need a long ciphertext in order to obtain reasonable frequencies. It is possible that at the first try we obtain some nonsense words but similar to some words of complete meaning, we can then change letters to correct these words. Repeating some times this step we would end up with the correct plaintext.

We have seen that we can easily break monoalphabetic ciphers applying this method, so the second property doesn't hold. Attacks to substitution ciphers are called **Statistical attacks**

Proof of the first property applied to a substitution cipher. We have to prove that:

$$\begin{aligned} x &= D_k(E_k(x)) && (1)\\ &= D_k(p(x)) && (2)\\ &= p^{-1}(p(x)) && (3)\\ &= x && (4)\\ & && (5) \end{aligned}$$

■

We have seen that monoalphabetic ciphers are prone to statistical attacks, since they preserve the statistical structure of the plaintext. A solution are the polyalphabetic ciphers in which the same symbol is not always mapped to the same encrypted symbol.

Polyalphabetic ciphers

An example of polyalphabetic cipher is the **Vigenére cipher** (XVI century). It works on "blocks" of m letters with a key of length m.

Example 17. The key is `FLUTE` (m=5). The plaintext is split into blocks of length 5 and the key `FLUTE` is repeated as necessary and used to encrypt each block.

THISISAVERYSECRETMESSAGE
+
FLUTEFLUTEFLUTEFLUTEFLUT
=
YSCLMXLPXVDDYVVJEGXWXLAX

Each letter of the ciphertext is given by the sum of the position of the letter in the plaintext plus the position of the letter in the key.

This type of ciphers works better than monoalphabetic ciphers because one letter is not always mapped to the same one unless they are at a distance that is multiple of m.

Formally, P=C=K=Z_{26}^m, where Z_{26}^m is $Z_{26}\times Z_{26}\times..\times Z_{26}$, m times.

- $E_{k1,..,km}(x_1,..,x_m) = (x_1+k_1,..,x_m+k_m)$ mod26;
- $D_{k1,..,km}(y_1,..,y_m) = (x_1-k_1,..,x_m-k_m)$ mod26.

If an attacker knows m, he has to try 26^m possible keys, if m is big enough, it is impossible to brute force it.

Lecture 3

Vigenére cipher cause an almost "flat" distribution of letters frequency, for this reason we would make many mistakes if we try do decrypt a ciphertext in the same way as we do for monoalphabetic ciphers.

As said in the last part of the previous lesson, if we want to brute force this cipher, (assuming we know m) we would have to try 26^m possible keys and that's infeasible, if we don't know m, it would we even worse. It is sufficient to choose m big enough to prevent brute force attacks, note that the number of keys grows exponentially with respect to the length.

Breaking Vigenére cipher

Even if the Vigenére cipher hides the statistic structure of the plaintext better than monoalphabetic ciphers, it still preserve most of it.

There are two famous methods to break this cipher, the first is due to Friedrick Kasiski (1863) and the second to Wolfe Friedman (1920). We will see the latter since it is more suitable to be mechanized. Both are based in **recover the length m of the key** and then **recover the key**.

The Friedman method uses statistical measures to recover the length m of the key. We consider the index of coincidence:

$$I_c(x) = \frac{\sum_{i=1}^{26} f_i(f_i - 1)}{n(n-1)} \approx \sum_{i=1}^{26} p_i^2 \tag{1}$$

where f_i is the frequency of the i-th letter in a text of length n, i.e., the number of times it occurs in such text and $p_i = f_i/n$ is the probability of the i-th letter. Intuitively, this measure gives the probability that two letters, chosen at random from the text, are the same. I compute this probability over all the letters.

Example 18. The IC of **"the index of coincidence"** is given by:
c(3*2) + d(2*1) + e(4*3) + f(1*0) + h(1*0) + i(3*2) + n(3*2) + o(2*1) + t(1*0) + x(1*0) = **34**
divided by n(n-1)=21*20 = **420**
which gives us an IC of 34/420 = **0.0809**

The IC of **"bmqvszfpjtcsswgwvjlio"** is given by:
b(1*0) + c(1*0) + f(1*0) + g(1*0) + i(1*0) + j(2*1) + l(1*0) + m(1*0) + o(1*0) + p(1*0) + q(1*0) + s(3*2) + t(1*0) + v(2*1) + w(2*1) + z(1*0) = **12**

divided by n(n-1)=21*20 = **420**
which gives us an IC of 12/420 = **0.0286**

Once I have the IC of my ciphertext, I have to compare it to the ICs of various languages to discover to which it corresponds.

The value of the index is maximum (value 1) for texts composed of just a single letter repeated n times. The value of the index is minimum (value $1/26 \approx 0.038$) for texts composed of letters chosen with uniform probability $1/26$.

The index of coincidence is thus a measure of how non uniformly letters are distributed in a text, each natural language has a characteristic index of coincidence, some examples:

$$\begin{aligned} \text{English} &\rightarrow 0.065 \\ \text{Russian} &\rightarrow 0.0529 \\ \text{German} &\rightarrow 0.0762 \\ \text{Spanish} &\rightarrow 0.0775 \end{aligned}$$

We can also use the Friedman method to find moro or polyalphabetic ciphers. We know that if we use frequencies analysis, if frequencies are flat, we have a polyalphabetic cipher, if we have peaks and valleys of frequencies, we have a monoalphabetic cipher. Considering the Friedman method, if the value of the index is minimum ≈ 0.038, we have a polyalphabetic cipher, if it is ≈ 0.065, we have a monoalphabetic cipher (same IC as English, just a permutation of letters).

With the Friedman method we can estimate m, the length of the key in a Vigenére cipher. The idea is to recover m by brute forcing, following this algorithm (in Python):

```
1  m=1
2  LIMIT = 0.06 #this is to check that ICs are above 0.06 and thus close to
                                        0.065 (assuming the text is in
                                        English)
3  found = False
4  while (not found):
5    sub = subciphers() #takes the m subciphertexts sub[m] obtained by
                                        selecting one letter every m
6    found = True
7    for i in range(0,m): #compute the IC of all subtexts
8      if IC(sub[i]) < LIMIT:
9        #if one of the IC is not as expected try to increase the length
10       found = False
11       m += 1
12       break
13 #survived the check, all ICs are above LIMIT
14 output (m)
```

It works because, once we reach the correct m, all the letters we are considering will be encrypted using the same key, "F" in the below example. So, computing the IC, we will obtain a value similar to the English IC value.

THISISAVERYSECRETMESSAGE
+
FLUTEFLUTEFLUTEFLUTEFLUT
=
YSCLMXLPXVDDYVVJEGXWXLAX

In order to obtain suitable results, we need to have a long enough ciphertext, otherwise we could not be able to succeed.

Now that we have m, we need to find the key. We already said that we cannot brute force it, it would be infeasible. What should we do?

- we divide the text into blocks of length m, as the length of the key (we just found it);
- we need to build new cryptograms with the first letter of each block, one with the second letter and so on;
- we analyse the new cryptograms as before and we find the shift in each position.

We are considering texts composed of letter at distance m from the first one, the second one, and so on. They have different shifts, we need to find the relative right shift.

The idea is to shift one subcipher until the mutual index of coincidence with the first subcipher becomes close to the one of the plaintext language, when this happens, we know that the applied shift is the relative shift between the two subciphers and , consequently, between the corresponding letters of the key.

The mutual index of coincidence is defined as:

$$MI_c(x, x') = \frac{\sum_{i=1}^{26} f_i f'_i}{nn'} = \sum_{i=1}^{26} p_i p'_i \tag{1}$$

It represents the probability that two letters taken from two texts x and x' are the same.

The following algorithm selects the relative shift that maximizes the mutual index of coincidence.

```
key = [] #empty list
for i in range(0,m): #for any letter of the key
  k = 0 #current relative shift
  mick = 0 #maximum index so far (we start with 0)
  for j in range(0,26): #for any possible relative shift
    #compute the mutual index of coincidence between the first subcipher
    #sub[0] and the i-th subcipher shifted by j
    mic = MIc(sub[0], shift(j,sub[i]))
    if mic > mick: #if it is the biggest so far
      k = j     #we remember the relative shift
      mick = mic  #.. and the maximum
  key.append(k)     #we append to the list the shift we have found
```

We repeat this for every letter of the key and we obtain the list of relative shifts, for example, if we obtain [0,4,6,3,9], it means that the second letter of the key is equal to the first plus 4, the third is equal to the first plus 6 and so on. But what is the first letter? The final step is to try all the possible 26 letter of the key, that gives us 26 possible keys.

Known-plaintext attacks

Until now we have considered attackers that only know the ciphertext y and try to find either the plaintext x or the key k. It is often the case that an attacker can guess part of the plaintext (e.g., the "standard" header of a message), if a message is split into blocks which are encrypted under the same key, it is reasonable to assume that an attacker can deduce part of the plaintext. For example if the attacker is trying to decrypt an email, he can guess the initial par that often starts with "Dear ...". If the attacker knows some plaintexts, this gives him the knowledge of some pairs (x,y) plaintext, ciphertext. Given this, the attacker should be able to decrypt other messages or to recover the key k (no matter which cipher is used).

The **Hill** cipher is a polyalphabetic cipher and it is a generalization of the Vigenére by introducing linear transformations of blocks of plaintext. We have P=C=Z_m^{26}, while K=$\{K|K$ is an invertible mod26 matrix m×m$\}$. The encryption and decryption are the following:

- $E_K(x_1,..,x_m) = (x_1,..,x_m)K$ mod26;
- $D_K(y_1,..,y_m) = (y_1,..,y_m)K^{-1}$ mod26.

Example 19. Let us assume M=(x_1,x_2)=(5,9) and K $=\begin{bmatrix}5 & 11\\ 8 & 3\end{bmatrix}$ (we will only consider 2×2 matrices for simplicity). Thus, $E_K(5,9) = (5,9) \times \begin{bmatrix}5 & 11\\ 8 & 3\end{bmatrix}$ mod26 $= (5 \times 5 + 9 \times 8, 5 \times 11 + 9 \times 3)$ mod26 $= (25 + 72, 55 + 27)$ mod26 $= (97, 82)$ mod 26 $= (19, 4)$

In order to decrypt a message, we need to compute the inverse of the matrix that is our key.

Example 20. We have (y_1,y_2) = (19,4) and K=$\begin{bmatrix}5 & 11\\ 8 & 3\end{bmatrix}$. To compute K^{-1}:

K^{-1} =$\det^{-1}$(K) $\begin{bmatrix}3 & -11\\ -8 & 5\end{bmatrix}$ mod26 = $\det^{-1}$(K)$\begin{bmatrix}3 & 15\\ 18 & 5\end{bmatrix}$ mod26.

We assume our matrix K is invertible. In the last step we changed the negative numbers to positive ones considering the mod26.

Now we can calculate det(K) $= (5 \times 3 - 11 \times 8)$mod26 $= (15 - 88)$mod26 = -73 mod26 = 5. How do we compute $\det^{-1}$(K)? To find the inverse mod26 of 5, we need to find a number in the interval [0,25] that multiplied by 5 mod26 gives 1. The number we are searching is 21, 5×21 mod26 = 105 mod26 = 1. Thus $\det^{-1}(K) = 21$.
Note that it is not always the case that the multiplicative inverse modulo exists, we will discuss this more in detail later on, introducing the public key cryptography and RSA.

Now we can solve $K^{-1} = \det^{-1}(K)\begin{bmatrix}3 & 15\\ 18 & 5\end{bmatrix}$ mod26= $21\begin{bmatrix}3 & 15\\ 18 & 5\end{bmatrix}$ mod26 $= \begin{bmatrix}63 & 315\\ 378 & 105\end{bmatrix}$ mod 26 $= \begin{bmatrix}11 & 3\\ 14 & 1\end{bmatrix}$. Thus $D_K(19,4) = (19,4)\begin{bmatrix}11 & 3\\ 14 & 1\end{bmatrix} = (19 \times 11 + 4 \times 14, 19 \times 3 + 4 \times 1)$ mod26 $= (265, 61)$ mod26 $= (5, 9)$.

Exercise 1. Encrypt and decrypt message (2,5) using a Hill cipher with K = $\begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix}$

Encryption

$E_K(2,5) = (2,5) \times \begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix} \text{mod} 26 = (2\times5+5\times8, 2\times11+5\times3) \text{mod} 26 = (10+40, 22+15) \text{mod} 26 = (50, 37) \text{ mod} 26 = (24, 11)$.

Decryption

$K^{-1} = \det^{-1}(K) \begin{bmatrix} 3 & -11 \\ -8 & 5 \end{bmatrix} \text{mod} 26 = \det^{-1}(K) \begin{bmatrix} 3 & 15 \\ 18 & 5 \end{bmatrix} \text{mod} 26.$

$\det(K) = (5 \times 3 - 11 \times 8) \text{mod} 26 = (15 - 88) \text{mod} 26 = -73 \text{ mod} 26 = 5.$

$\det^{-1}(K) = 21,\ 5 \times 21 \text{ mod} 26 = 105 \text{ mod} 26 = 1.$

$K^{-1} = \det^{-1}(K) \begin{bmatrix} 3 & 15 \\ 18 & 5 \end{bmatrix} \text{mod} 26 = 21 \begin{bmatrix} 3 & 15 \\ 18 & 5 \end{bmatrix} \text{mod} 26 = \begin{bmatrix} 63 & 315 \\ 378 & 105 \end{bmatrix} \text{mod } 26 = \begin{bmatrix} 11 & 3 \\ 14 & 1 \end{bmatrix}.$

$D_K(24,11) = (24,11) \begin{bmatrix} 11 & 3 \\ 14 & 1 \end{bmatrix} = (24 \times 11 + 11 \times 14, 24 \times 3 + 11 \times 1) \text{ mod} 26 = (418, 83) \text{ mod} 26 = (2, 5)$.

Lecture 4

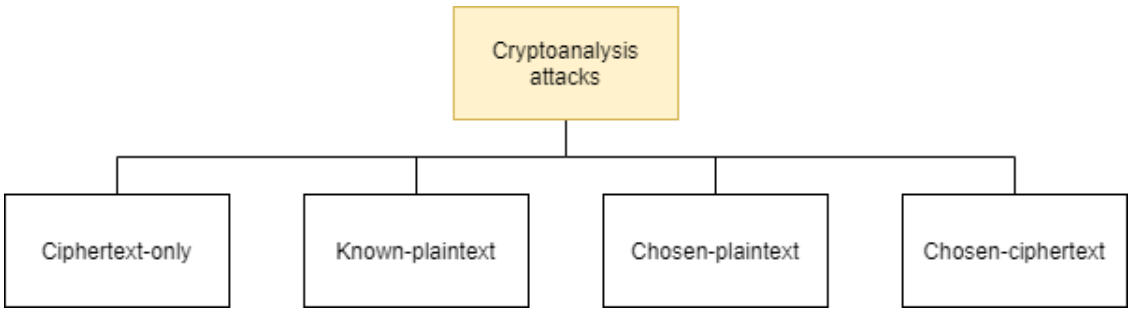

Figure 10: Types of Cryptonalysis attacks.

Up to now we have seen **ciphertext-only** attacks and **known-plaintext** attacks (in the last part of the previous lecture).
In a ciphertext-only attack, the attacker is assumed to have access only to a set of ciphertexts (e.g., monoalphabetic ciphers, polyalphabetic ciphers), the limit is that it is easy to find the correspondence between letters in the plaintext and in the ciphertext.
In a known-plaintext attack, the attacker knows some pairs (x',y'), (x",y"), .. of plaintexts/ciphertexts (e.g., Hill ciphers), the limit in this case is that it is a linear transformation of plaintext block into a cipher block.

A chosen-plaintext attack (CPA) is an attack model for cryptoanalysis which presumes that the attacker can ask and obtain the ciphertexts for given plaintexts.
A chosen-ciphertext attack (CCA) is an attack model for cryptoanalysis where the cryptoanalyst can gather information by obtaining the decryption of chosen ciphertexts.

Exercise 2. Encrypt and decrypt message (1,3) using a Hill cipher with K = $\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$

Encryption

$E_K(1,3) = (1,3) \times \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} \text{ mod} 26 = (1 \times 1 + 3 \times 4, 1 \times 2 + 3 \times 3) \text{ mod} 26 = (1 + 12, 2 + 9) \text{ mod} 26 = (13, 11) \text{ mod} 26 = (13, 11).$

Decryption

$K^{-1} = \det^{-1}(K) \begin{bmatrix} 3 & -2 \\ -4 & 1 \end{bmatrix} \text{ mod} 26 = \det^{-1}(K) \begin{bmatrix} 3 & 24 \\ 22 & 1 \end{bmatrix} \text{ mod} 26.$

$\det(K) = (1 \times 3 - 2 \times 4) \text{mod} 26 = (3 - 8) \text{mod} 26 = -5 \text{ mod} 26 = 21.$

$\det^{-1}(\mathrm{K}) = 5$, $21 \times 5 \bmod 26 = 105 \bmod 26 = 1$.
$\mathrm{K}^{-1} = \det^{-1}(\mathrm{K}) \begin{bmatrix} 3 & 24 \\ 22 & 1 \end{bmatrix} \bmod 26 = 5 \begin{bmatrix} 3 & 24 \\ 22 & 1 \end{bmatrix} \bmod 26 = \begin{bmatrix} 15 & 120 \\ 110 & 5 \end{bmatrix} \bmod 26 = \begin{bmatrix} 15 & 16 \\ 6 & 5 \end{bmatrix}$.
$\mathrm{D}_K(13,11) = (13,11) \begin{bmatrix} 15 & 16 \\ 6 & 5 \end{bmatrix} = (13 \times 15 + 11 \times 6, 13 \times 16 + 11 \times 5) \bmod 26 = (261, 263) \bmod 26 = (1, 3)$.

Attack to the Hill cipher

If I am an attacker and I have a specific number of pairs (plaintext, ciphertext), I can attack the Hill cipher. With a m×m matrix, I need m pairs, if I have a 2×2 matrix, I will need 2 pairs (plaintext, ciphertext) to be able to extract the key. Once I have the key, I can decrypt all the messages that will be sent later on.

We know that:

$$(y_1^1, ..., y_m^1) = (x_1^1, ..., x_m^1)\mathrm{K} \bmod 26$$
$$...$$
$$(y_1^m, ..., y_m^m) = (x_1^m, ..., x_m^m)\mathrm{K} \bmod 26$$

which can be written as $\mathrm{Y} = \mathrm{XK} \bmod 26$ with:

$$\mathrm{X} = \begin{bmatrix} x_1^1 & .. & x_m^1 \\ .. & .. & .. \\ x_1^m & .. & x_m^m \end{bmatrix} \text{ and } \mathrm{Y} = \begin{bmatrix} y_1^1 & .. & y_m^1 \\ .. & .. & .. \\ y_1^m & .. & y_m^m \end{bmatrix}$$

If X^{-1} exists, we can write $\mathrm{X}^{-1}\mathrm{Y} \bmod 26 = \mathrm{X}^{-1}\mathrm{XK} \bmod 26$ and then $\mathrm{K} = \mathrm{X}^{-1}\mathrm{Y} \bmod 26$.

Example 21. Let's define:

$$\mathrm{x}_1{=}(5,9) \rightarrow \mathrm{y}_1{=}(19,4)$$
$$\mathrm{x}_2{=}(2,5) \rightarrow \mathrm{y}_2{=}(24,11)$$

We can write $(19,4) = (5,9)\mathrm{K} \bmod 26$ and $(24,11) = (2,5)\mathrm{K} \bmod 26$.
We can construct X and Y matrices as follows:

$$\mathrm{X} = \begin{bmatrix} 5 & 9 \\ 2 & 5 \end{bmatrix} \text{ and } \mathrm{Y} = \begin{bmatrix} 19 & 4 \\ 24 & 11 \end{bmatrix}.$$

Now, if X^{-1} exists, we can recover the key K. We compute X^{-1} in the same way we did previously for K^{-1}.
$\mathrm{X}^{-1} = \det^{-1}(\mathrm{X}) \begin{bmatrix} 5 & -9 \\ -2 & 5 \end{bmatrix} \bmod 26 = \det^{-1}(\mathrm{X}) \begin{bmatrix} 5 & 17 \\ 24 & 5 \end{bmatrix} \bmod 26.$

$\det(\mathrm{X}) = (5 \times 5 - 9 \times 2) = (25\text{-}18) = 7$.
$\det^{-1}(\mathrm{X})$ is a number a such that $7{\times}a = 1 \bmod 26$, this number $a = 15$.

Thus $\mathrm{X}^{-1} = \det^{-1}(\mathrm{X}) \begin{bmatrix} 5 & 17 \\ 24 & 5 \end{bmatrix} \bmod 26 = 15 \begin{bmatrix} 5 & 17 \\ 24 & 5 \end{bmatrix} \bmod 26 = \begin{bmatrix} 75 & 255 \\ 360 & 75 \end{bmatrix} \bmod 26 = \begin{bmatrix} 23 & 21 \\ 22 & 23 \end{bmatrix}$.

Then, $K = X^{-1}Y \bmod 26 = \begin{bmatrix} 23 & 21 \\ 22 & 23 \end{bmatrix} \begin{bmatrix} 19 & 4 \\ 24 & 11 \end{bmatrix} \text{mod}26 = \begin{bmatrix} 23 \times 19 + 21 \times 24 & 23 \times 4 + 21 \times 11 \\ 22 \times 19 + 23 \times 24 & 22 \times 4 + 23 \times 11 \end{bmatrix}$

$\text{mod}26 = \begin{bmatrix} 941 & 323 \\ 970 & 341 \end{bmatrix} \bmod 26 = \begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix}$.

So, our key $K = \begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix}$.

Modern ciphers always contain a non-linear component to prevent this kind of attacks.

If th matrix X is not invertible, if the attacker has no more pairs of (plaintext, ciphertext) he lose, otherwise he can try constructing other X matrices that maybe are invertible.

Block ciphers

Block ciphers are cryptosystems that "reuse" the same key to encrypt letters or blocks of the plaintext. For example shift ciphers are part of block ciphers. They are weak because if an attacker know that it is always used the same key, he could recover the key and decrypt al the messages.

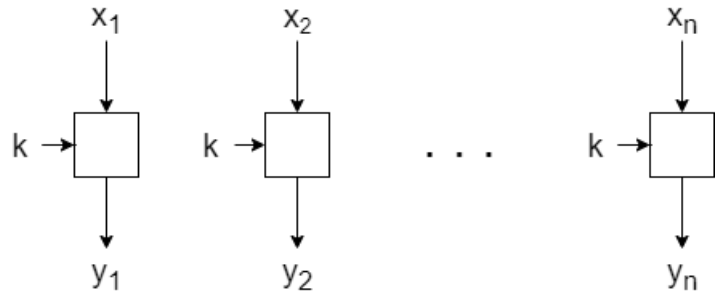

Figure 11: Example of block ciphers.

Stream ciphers

Definition 10. **Stream ciphers** are cryptosystems that use a stream of key $z_1, z_2, ..., z_n$ instead of a single one.

The idea is to encrypt the first letter of the plaintext with z_1, the second with z_2 and so on, it doesn't matter if we encrypt a letter or a block of text, what matters is that the key used is always different.

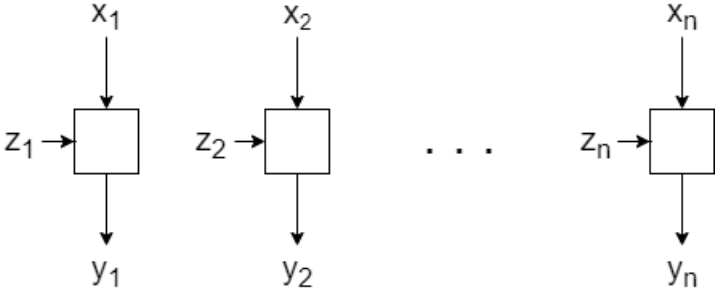

Figure 12: Example of stream ciphers.

Having a different key for each letter or block of the plaintext is of course appealing but not much practical. The stream of keys is thus usually derived starting from an initial key k, but it can also depend on previous parts of the plaintext. In general we say that

$$z_i = f_i(k,x_i, ..., x_{i-1}).$$

That means that the i-th key depends on k and on the previous i-1 letters or blocks. So $z_1 = f_1(k)$, we can compute it without knowledge of the plaintext, to compute z_2, instead, we need to know x_1 because $z_2 = f_2(k,x_1)$. The values are thus computed in the following sequence: z_1, x_1, z_2, x_2, ...
This slow down a bit the procedure because I can't do anything in parallel.

Block ciphers are a subset of stream ciphers in which i have z_i=k for each i.

A stream cipher is **periodic** if its key stream has the form:

$$z_1, z_2, .., z_d, z_1, z_2, .., z_d, z_1, ...$$

So if it repeats after **d** steps. This form of cipher reminds us to the **Vigenére cipher**. It can be seen as a stream cipher acting on single letters and with a periodic key stream.

Exercise 3. Formalize the cipher giving (P,C,K,E,D) and defining the key stream z_i.

- P=C=K=Z_{26};
- $E_{zi}(x_i) = (x_i+z_i)$ mod26;
- $D_{zi}(y_i) = (y_i-z_i)$ mod26;
- $z_i = k_{(i \bmod m)}$.

A stream cipher is **synchronous** if its key stream does not depend on the plaintexts (for example $z_i = f_i(k)$ for all i). So the key stream can be generated starting from k and independently on the plaintext. It is useful to improve efficiency because we do not need to obtain x_i to compute z_{i+1}, so the key stream can be generated offline, before the actual ciphertext is received. We can consider Caesar cipher Hill cipher and also Vigenére cipher part of synchronous ciphers.

Asynchronous stream ciphers

In general asynchronous stream ciphers generate keys that depends either from k and from the previous plaintexts:

$$z_i = f_i(k, x_1, ..., xi-1)$$

This means that, if we are decrypting, we need to decrypt and, at the same time, compute the keys stream as a key can depend on previous plaintexts. An example is the **Autokey cipher**. We define it in the same way of a shift cipher, so P=C=K=Z_{26} and

- $E_{zi}(x_i) = (x_i + z_i)$ mod26;
- $D_{zi}(y_i) = (y_i - z_i)$ mod26.

We define the key stream as:

$$z_i = \begin{cases} k & \text{if i=0} \\ x_{i-1} & \text{if i}\geq 2 \end{cases}$$

The first key is the initial key k and the next keys are the same as the previous plaintext.

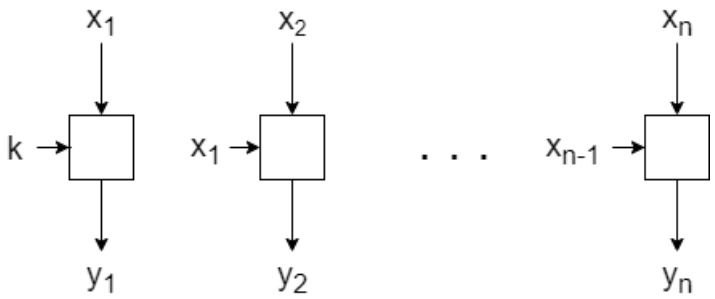

Figure 13: Encryption in an autokey cipher.

Exercise 4. Let's assume the plaintext is the word "networksecurity", what is the encryption using k=5?
First of all we need to substitute our word with the corresponding position in the alphabet and it is: "13 4 19 22 14 17 10 18 4 2 20 17 8 19 24". We will encrypt our string as follows:

E(13) = (13+5) mod26 = 18
E(4) = (4+13) mod26 = 17
...
E(24) = (24+19) mod26 = 17

At this point we just need to transform back our numbers into letters and we obtain the ciphertext.

Exercise 5. Try to extract the plaintext from this word encoded using Autokey. We do not know the key k.

`FTPNIH`

Since we do not know the key k, we have to try all the 26 possible k to see which decryption produces a meaningful word.

- k=0 will produce `FOBMWL`, nonsense;
- k=1 will produce `EPANVM`, nonsense;
- k=2 will produce `DQZOUN`, nonsense;
- k=3 will produce `CRYPTO`, it has sense!

Exercise 6. Suppose that we know that `FRIDAY` has been encrypted as `PQCFKU` using the Hill cipher, What is K? Assume K is a 2×2 matrix.

We can consider the pairs:
`(F,R)` → `(P,Q)` = `(5,17)` → `(15,16)`
`(I,D)` → `(C,F)` = `(8,3)` → `(2,5)`
`(A,Y)` → `(K,U)` = `(0,24)` → `(10,20)`

We can construct X and Y matrices as follows:

$$\mathrm{X} = \begin{bmatrix} 5 & 17 \\ 8 & 3 \end{bmatrix} \text{ and } \mathrm{Y} = \begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix}.$$

Now, if X^{-1} exists, we can recover the key K. We compute X^{-1} in the same way we did previously for K^{-1}.

$\mathrm{X}^{-1} = \det^{-1}(\mathrm{X}) \begin{bmatrix} 3 & -17 \\ -8 & 5 \end{bmatrix} \text{mod} 26 = \det^{-1}(\mathrm{X}) \begin{bmatrix} 3 & 9 \\ 18 & 5 \end{bmatrix} \text{mod} 26.$

$\det(\mathrm{X}) = (5 \times 3 - 17 \times 8) = (15\text{-}136) =$ -121 mod26 = -17 mod26 = 9.
$\det^{-1}(\mathrm{X})$ is a number a such that $9 \times a = 1$ mod26, this number is $a = 3$.

$$\text{Thus } \mathrm{X}^{-1} = \det^{-1}(\mathrm{X}) \begin{bmatrix} 3 & 9 \\ 18 & 5 \end{bmatrix} \text{mod} 26 = 3 \begin{bmatrix} 3 & 9 \\ 18 & 5 \end{bmatrix} \text{mod} 26 = \begin{bmatrix} 9 & 27 \\ 54 & 15 \end{bmatrix} \text{mod} 26 = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix}.$$

$$\text{Then, } \mathrm{K} = \mathrm{X}^{-1}\mathrm{Y} \bmod 26 = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix} \begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix} \text{mod} 26 = \begin{bmatrix} 9 \times 15 + 1 \times 2 & 9 \times 16 + 1 \times 5 \\ 2 \times 15 + 15 \times 2 & 2 \times 16 + 15 \times 5 \end{bmatrix}$$

$$\text{mod} 26 = \begin{bmatrix} 137 & 149 \\ 60 & 107 \end{bmatrix} \bmod 26 = \begin{bmatrix} 7 & 19 \\ 8 & 3 \end{bmatrix}.$$

$$\text{So, our key } \mathrm{K} = \begin{bmatrix} 7 & 19 \\ 8 & 3 \end{bmatrix}.$$

Lecture 5

Definition 11. A **perfect cipher** is a cipher that can never be broken, even after an unlimited time.

But do perfect ciphers exist? Yes, but they are only theoretically (they can be implemented), in practice they are unpractical.

The theory behind perfect ciphers has been developed by **Claude Shannon**. It assumes an only-ciphertext model of the attacker, this means that the attacker only knows the ciphertext y and tries to find the plaintext x or the key k.

We can provide another definition for perfect ciphers:

Definition 12. A cipher system is said to offer **perfect secrecy** if, on seeing the ciphertext, the interceptor gets no extra information about the plaintext than he had before the ciphertext was observed. In a cipher system with perfect secrecy, the interceptor is forced to guess the plaintext.

To formalize them we need to introduce a bit of probability:

- $\mathrm{p}_p(\mathrm{x})$ is the probability of plaintext x to occur;
- $\mathrm{p}_K(\mathrm{k})$ is the probability of certain key k to be used as encryption key.

Given a plaintext and a key there exists a unique corresponding ciphertext.

The two probability distributions $\mathrm{p}_p(\mathrm{x})$ and $\mathrm{p}_K(\mathrm{k})$ induce a probability distribution on the ciphertexts.

$$p_c(y) = \sum_{k \in K, \exists x | E_k(x) = y} p_K(k) \times p_p(D_k(y)) \tag{1}$$

This means that, given a ciphertext y, we look for all the keys that can give such a ciphertext from some plaintext x. Then we sum the probability of all such keys times the probability of the corresponding plaintext.

Example 22. Let's consider the following toy-cipher with P=a,b, K=k_1,k_2, C1,2,3. The encryption is defined in the following table:

E	**a**	**b**
$\mathbf{k}_1$	1	2
$\mathbf{k}_2$	2	3

Let $\mathrm{p}_p(\mathrm{a})=3/4$, $\mathrm{p}_p(\mathrm{b})=1/4$, $\mathrm{p}_K(\mathrm{k}_1)=\mathrm{p}_K(\mathrm{k}_2)=1/2$.

Now we want to compute $p_C(1)$, $p_C(2)$, $p_C(3)$.

$p_C(1) = p_p(a) \times p_K(k_1) = 3/4 \times 1/2 = 3/8$
$p_C(2) = p_p(a) \times p_K(k_2) + p_p(b) \times p_K(k_1) = 3/4 \times 1/2 + 1/4 \times 1/2 = 3/8 + 1/8 = 1/2$
$p_C(3) = p_p(b) \times p_K(k_2) = 1/4 \times 1/2 = 1/8$

Definition 13. The **conditional probability of a ciphertext** y with respect to a plaintext x, computes how likely is a certain y once we fix x.

$$p_c(y|x) = \sum_{k \in K, E_K(x)=y} p_K(x) \tag{1}$$

More simply is just the sum of the probability of all keys giving y from x.

Example 23. The conditional probability of ciphertext 1 with respect to the two plaintexts a and b is:

$$p_c(1|a) = \sum_{k \in K, E_K(x)=y} p_K(x) = p_K(k_1) = 1/2 \tag{1}$$

$$p_c(1|b) = \sum_{k \in K, E_K(x)=y} p_K(x) = 0 \tag{2}$$

We can notice that 1 can never be obtained from b. Furthermore we can see that if an attacker sees a 1, he can deduce that it is an a, this means that this is not a perfect cipher.

The conditional probability of a plaintext with respect to a ciphertext is related to the security of the cipher. It is a measure of how likely is a plaintext once a ciphertext is observed (which is what the attacker is usually interested to know).

With Pr(y)>0, we can define the **Bayes' Theorem** as:

$$Pr(x|y) = \frac{Pr(x) \times Pr(y|x)}{Pr(y)} \tag{1}$$

We can apply this theorem to our case to get the conditional probability of a plaintext x with respect to a ciphertext y:

$$p_p(x|y) = \frac{p_p(x) \times p_c(y|x)}{p_c(y)} \tag{1}$$

Example 24. At this point we can compute $p_p(a|1)$ and $p_p(b|1)$ as follows:

$$p_p(a|1) = \frac{p_p(a) \times p_c(1|a)}{p_c(1)} = \frac{3/4 \times 1/2}{3/8} = 1 \tag{1}$$

$$p_p(b|1) = \frac{p_p(b) \times p_c(1|b)}{p_c(1)} = \frac{1/4 \times 0}{3/8} = 0 \tag{2}$$

Thus, when observing a 1 as a result of a conditional probability of a plaintext x with respect to a ciphertext y, we are sure it is plaintext x, and this means that the cipher is completely insecure.

Exercise 7. What will be the probabilities of plaintexts a and b with respect to ciphertext 3?

First of all we need to find the conditional probabilities of ciphertext 3 with respect to the plaintexts a and b:

$$p_c(3|a) = \sum_{k \in K, E_K(x)=y} p_K(x) = 0 \tag{1}$$

$$p_c(3|b) = \sum_{k \in K, E_K(x)=y} p_K(x) = p_K(k_2) = 1/2 \tag{2}$$

We can now compute the conditional probabilities of plaintexts a and b with respect to the ciphertext 3:

$$p_p(a|3) = \frac{p_p(a) \times p_c(3|a)}{p_c(3)} = \frac{3/4 \times 0}{1/8} = 0 \tag{1}$$

$$p_p(b|3) = \frac{p_p(b) \times p_c(3|b)}{p_c(3)} = \frac{1/4 \times 1/2}{1/8} = 1 \tag{2}$$

Exercise 8. What if we compute the probabilities of a and b when observing 2?

First of all we need to find the conditional probabilities of ciphertext 2 with respect to the plaintexts a and b:

$$p_c(2|a) = \sum_{k \in K, E_K(x)=y} p_K(x) = p_K(k_1) = 1/2 \tag{1}$$

$$p_c(2|b) = \sum_{k \in K, E_K(x)=y} p_K(x) = p_K(k_2) = 1/2 \tag{2}$$

We can now compute the conditional probabilities of plaintexts a and b with respect to the ciphertext 2:

$$p_p(a|2) = \frac{p_p(a) \times p_c(2|a)}{p_c(2)} = \frac{3/4 \times 1/2}{1/2} = 3/4 \tag{1}$$

$$p_p(b|2) = \frac{p_p(b) \times p_c(2|b)}{p_c(2)} = \frac{1/4 \times 1/2}{1/2} = 1/4 \tag{2}$$

At this point we can provide an extra definition for perfect ciphers:

Definition 14. A cipher is perfect iff $p_p(x|y)=p_p(x)$ for all x in P and for all y in C.

Intuitively, a cipher is perfect if observing a ciphertext y gives no information about any of the possible plaintexts x.

The cipher in the example is far from being perfect, but it satisfies the above definition for ciphertext 2.

A cipher is perfect if there is some key that maps any message to any ciphertext with equal probability.

Lecture 6

To prove that a cipher is not perfect, we just need to find a x in P and a y in C that does not satisfy definition 14

Exercise 9. Prove that the shift cipher with $p_K(k) = 1/|K| = 1/26$ (i.e., with keys picked at random for each letter of the plaintext), is a perfect cipher. If we change key at any time we encrypt a letter, the shift cipher becomes perfect (unbreakable).

We can define the cipher as follows:

- P=C=K=Z_{26};
- $E_k(x) = (x+k) \bmod 26$;
- $D_k(y) = (y-k) \bmod 26$;
- k changes at each encryption.

We want to prove that definition 14 holds for this cipher.

We compute the probability of a generic ciphertext y as:

$$p_C(y) = \sum_{k \in K, \exists x. E_k(x)=y} p_K(k) \times p_p(D_k(y)) \tag{1}$$

$$= \frac{1}{26} \sum_{k \in K, \exists x. E_k(x)=y} p_p(D_k(y)) \tag{2}$$

$$= \frac{1}{26} \sum_{k \in K} p_p((y-k) mod 26) \tag{3}$$

$$= \frac{1}{26} \sum_{x \in P} p_p(x) = \frac{1}{26} \tag{4}$$

Note that for each key k, we always have the plaintext (y-k) mod26 that gives y when encrypted under k, and that for all possible keys gives all possible plaintexts x and sums to 1.

$$p_c(y|x) = \sum_{k \in K, E_k(x)=y} p_K(k) \tag{1}$$

$$= p_K((y-x) mod 26) = \frac{1}{26} \tag{2}$$

We get that k = (y-x) mod26 from the decryption formula, given x and y, there exists a unique key k that encrypts x as y and it is (y-x)mod26.

Finally we can say that:

$$p_p(x|y) = \frac{p_p(x) \times p_c(y|x)}{p_c(y)} \tag{1}$$

$$= \frac{p_p(x) \times \frac{1}{26}}{\frac{1}{26}} = p_p(x) \tag{2}$$

We have demonstrated that $p_p(y|x) = p_p(x)$ for all x in P and for all y in C, thus the cipher is perfect.

Theorem 1. Let $p_c(y) > 0$ for all y. A cipher is perfect only if |K|≥|P|.

This means that a necessary condition for a cipher to be perfect is that the number of keys is at least the same as the number of plaintexts.

Proof. For Bayes' theorem:

$$p_p(x|y) = \frac{p_p(x) \times p_c(y|x)}{p_c(y)}$$

Thus, given $p_p(x|y) = p_p(x)$ (the cipher is perfect), we have $p_c(y|x) = p_c(y)$ (these two quantities erase) for all x in P and for all y in C.

As an assumption $p_c(y) > 0$.
If we fix x, we obtain that for each y, $p_c(y|x) = p_c(y) > 0$. This means that there exists at least one key k such that E_k(x)=y (otherwise $p_c(y|x) = 0$).

All such keys are different since E_k is a function and we have fixed x, and x cannot be mapped to two different ciphertexts by the same key. Thus, we have at least one key for each ciphertext (|K|≥|C|).

Since, for any cipher (not necessarily perfect), E_k injects the set of plaintexts into the set of ciphertext, we also have |C|≥|P|, thus |K|≥|C|≥|P|, i.e.,

$$\mathbf{|K| \geq |P|}$$

■

Exercise 10. Prove that the ciphertext defined as $E_k(x_1,...,x_d)=(x_1+k,...,x_d+k)$mod26 is not perfect.

We can prove this in two ways:

1. using the theorem 1, proving that |K|<|P|.
   Our set K={0, 1, 2, .., 25}, so |K|=26, we need to find |P|. Since x=$\{x_1,..,x_d\}$ and each x_i can assume 26 values, we have that $|P|=26^d$. Thus $26<26^d$, so |K|<|P| (with d≥2).

2. using definition 14. Let's assume x=$\{x_1,..,x_d\}$ is an English word and let's also assume d=5. We are looking for an English word of length 5, for example *GOOFY*. Since it is a real word, p_p(GOOFY)>0. We want to find a y such that p_p(GOOFY|y)$\neq$ p_p(GOOFY)>0. For example with y=AAAAA, p_p(GOOFY|AAAAA)=0 and we are done. We could have used other y, such as y=ABCDE.

Theorem 2. Let |P|=|C|=|K|. A cipher is perfect iff

1. $p_K(k) = 1/|K|$ for all k in K;
2. for each x in P and y in C, there exists exactly one key k such that E_k(x)=y.

The theorem states that, for a cipher to be perfect (given that the size of P, C and K is the same), keys should be picked at random for any encryption and each plaintext is mapped into each ciphertext through a unique key.

Proof. We will prove the $\rightarrow$ direction of the theorem.

For theorem 1, in a perfect cipher |K|$\geq$|C| if p_c(y)>0 for all y. If we fix x, we obtain that for each y p_c(y|x)=p_c(y)>0, i.e., there exists at least one key k such that E_k(x)=y and all of the other keys are different.

Here (by assumption) |K|=|C|, meaning that all of these keys k are unique (otherwise we would have |K|>|C|). Since this holds for each x and y, we have proved condition 2, i.e., that for each x in P and y in C, there exists exactly one key k such that E_k(x)=y.

To prove condition 1, it is enough to notice that given

$$p_c(y|x) = \sum_{k \in K, E_k(x)=y} p_K(k)$$

p_c(y|x)=p_p(y), i.e., the probability of y given x is equal to the probability of the unique key k that encrypts x into y.

Thus, p_K(k)=p_c(y|x)=p_c(y).
If we fix y and we consider all possible plaintexts x we obtain all possible keys k and for all of them it holds p_K(k)=p_c(y). Since the number of keys is |K|, and since they all have the same probability (p_c(y)), and given that the sum of the probabilities of all keys must be 1 and |P|=|C|=|K|, we obtain condition 1, p_K(k)=1/|K|.

■

We can use this theorem to prove that a cipher is (or is not) perfect.

The one time pad

This perfect cipher has been used for the telegraph and is a binary variant of Vigenére with keys picked at random.

More precisely we have P=C=K=Z_{2^d} with p_K(k)=1/|K|=$1/2^d$ for all k in K. 2^d means sequences of binary numbers of length d.

The encryption function is the following: $E_{k_1,...,k_d}(x_1,...,x_d)$ = (x_1 xor k_1,...,x_d xor k_d), where xor is a bitwise xor operation.

Is it a perfect cipher? Condition 2 of theorem 2 is satisfied by definition. Condition 2 holds too because we are using xor operations.

Lesson learned

Shannon theory on perfect ciphers shows that they exist but require as many keys as the possible plaintexts, and keys need to be picked at random for each encryption.

Even if this makes such ciphers unpractical, the one-time-pad has been used for real transmission. The setup consisted of two identical books with thousands of "random" keys. Each key was used only once. Once the book has been used completely, new shared books were necessary.