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).

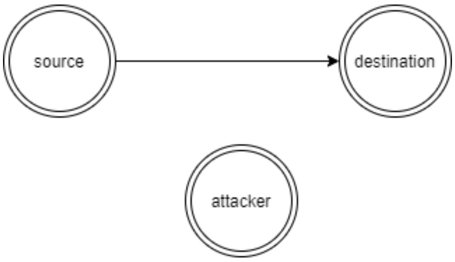

Figure 1: Expected information flow

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

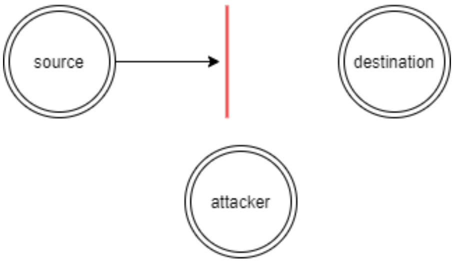

Figure 2: The attacker interrupts the flow of information

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

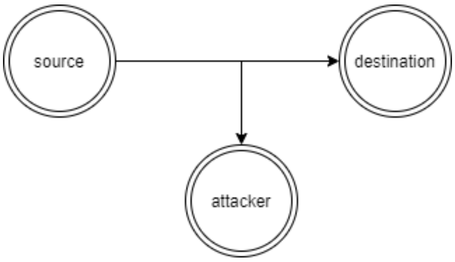

Figure 3: The attacker intercepts information

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.

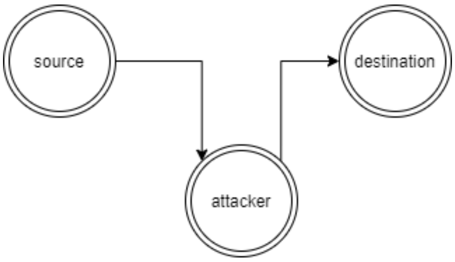

Figure 4: The attacker modifies information

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.

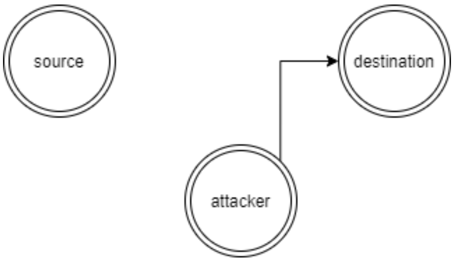

Figure 5: The attacker forges new information

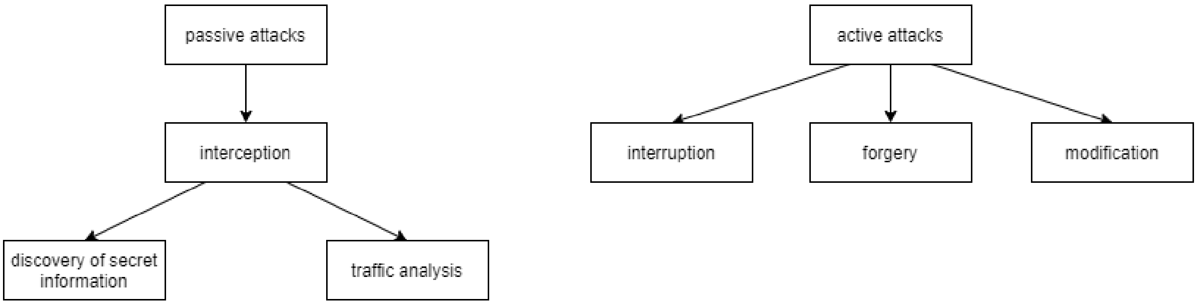

Figure 6: Classification of different types of attacks

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

`A → 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.

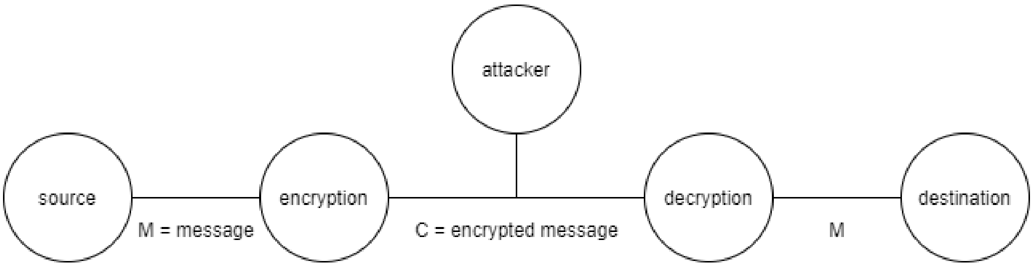

Figure 7: Encryption with a shared key

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

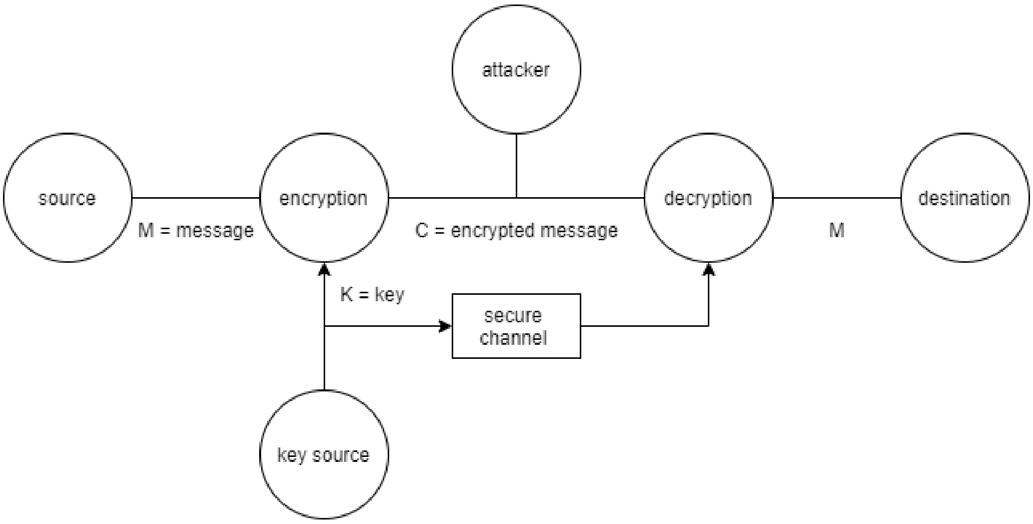

Figure 8: Encryption with a shared key and the secure channel

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)

$$\begin{array}{c} \texttt{A} \rightarrow \texttt{D} \\ \texttt{B} \rightarrow \texttt{E} \\ \texttt{C} \rightarrow \texttt{F} \\ \dots \\ \texttt{Z} \rightarrow \texttt{C} \end{array}$$

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:

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

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:

$$\text{E}_k(x) = (x+k)\text{mod}26$$
$$\text{D}_k(y) = (y-k)\text{mod}26$$

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:

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

■

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

Definition 9. A **group** $\langle \text{G}, * \rangle$ 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 $\langle \text{G}, * \rangle$);
- 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 $\langle \text{Z}, + \rangle$, 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 $\langle \text{Z}, + \rangle$);
- 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 $\langle \text{Z}, \cdot \rangle$, 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.

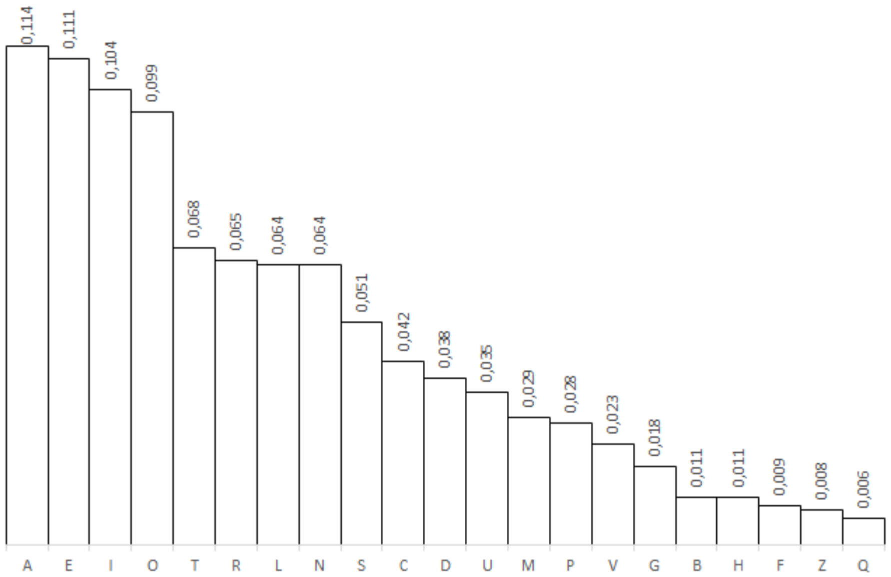

Figure 9: Percentage frequencies of letters in Italian language.

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.