Université Libre de Bruxelles
Faculté des Sciences

2022-2023

INFO-F-405: Introduction to cryptography

# 1. Historical ciphers and general principles

## Historical ciphers

### Exercise 1

A cipher with $\mathcal{M} = \mathcal{C}$ for certain keys $k \in \mathcal{K}$ is called *involutory* if its encryption and decryption procedures become identical, i.e., for all $m \in \mathcal{M}$, $E_k(m) = D_k(m)$. Under which keys $k \in [0, 25]$ the shift encryption scheme (see slides) becomes involutory?

**Answer of exercise 1**

For $k = 13$ and $k = 26$. For instance, for $k = 13$ plaintext letter `A` will be encrypted to the ciphertext letter `N`, which is equivalent to computing $14 = 1 + 13 \pmod{26}$ whereas the decryption will compute $1 = 14 + 13 \pmod{26}$. For $k = 13$ the Shift Cipher is sometimes called ROT13. Note that if $k = 26$ the encryption procedure becomes an identity function, i.e. $\#c_i = \#m_i$. That is, using $k = 26$ is not interesting since the ciphertext is identical to the plaintext.

### Exercise 2

The following text was encrypted using the shift encryption scheme. Try all possible secret keys to find the one that decrypts the following message:

`Fbzr crbcyr jrne Fhcrezna cnwnznf. Fhcrezna jrnef Puhpx Abeevf cnwnznf.`

How many attempts are needed to be sure to find the correct one?

Assuming the secret key is a uniformly-distributed random variable. What is the probability that the correct key is found after $t$ attempts?

Here is another ciphertext to decrypt:

`'Yvccf, nficu!' - zj fev fw kyv wzijk kyzexj gvfgcv kip kf gizek nyve kyvp cvrie r evn gifxirddzex crexlrxv.`

### Answer of exercise 2

The first ciphertext decrypts to `Some people wear Superman pajamas. Superman wears Chuck Norris pajamas.` The secret key is 13.

In this case, exhaustive key search takes up to 26 attempts to find the correct key. If the key is uniformly distributed, the probability to find it after $0 \leq t \leq 26$ attempts is $\frac{t}{26}$.

The second ciphertext decrypts to `'Hello, world!' - is one of the first things people try to print when they learn a new programming language.` The secret key is 17.

## Exercise 3

In one of the previous exercices, we performed cryptanalysis by knowing only the ciphertexts. Here we consider known-plaintext attacks where the attacker gets access to some know plaintext/ciphertext pairs and wants to determine the secret key to be able to decrypt past or future messages.

How many pairs of plaintext/ciphertext characters (or bits) must be known in order to determine the secret key without ambiguity in the following ciphers?

1. In the shift encryption scheme?
2. In a general mono-alphabetic substitution scheme?
3. In a general poly-alphabetic substitution scheme (assuming the length $t$ of the key is known)?
4. In the Vigenère cipher ($t$ is known)?
5. In the binary Vigenère cipher ($t$ is known)?

### Answer of exercise 3

1. One pair $(m_i, c_i)$ is enough since the same key $k = \#m_i - \#c_i \pmod{26}$ is used to encrypt all characters.
2. To remove any ambiguity, the attacker should collect enough pairs $(m_i, c_i)$ until $c_i$ takes 25 different letter values. This determines the permutation $k$, as the mapping for the 26-th letter can be deduced from the 25 others.

3. This follows the same idea as for the mono-alphabetic substitution, except that the collection must be done independently for each position $i$ modulo $t$. E.g., if $t = 2$, the attacker must collect enough pairs $(m_i, c_i)$ until $c_i$ takes 25 different letter values for odd $i$'s, and the same for even $i$'s.

4. Since, in the Vigenère cipher, the key $k$ consists of $t$ characters that are periodically used to encrypt the plaintext, the analyst must known first $t$ characters of the plaintext/ciphertext pair $(m, c)$. Then each of these $t$ characters can be found as in the shift encryption scheme.

5. Similarly, a pair of at least $t$ bits must be known. The key is found by bitwise modulo-2 adding the plaintext and the ciphertext.

# Perfect secrecy vs computational security

## Exercise 4

A bank defines an encryption algorithm to encrypt their account numbers. An account number is a 12-digit number, i.e., an element of $\mathbb{Z}_{10^{12}}$. Every time an account number is encrypted, a fresh key is chosen randomly and uniformly from the same set $\mathbb{Z}_{10^{12}}$. The encryption goes as follows:

$$E_k(m) = m + k \pmod{10^{12}} \quad D_k(c) = c - k \pmod{10^{12}}$$

Does this cipher satisfy perfect secrecy?

**Answer of exercise 4**

Yes, as only one key maps a given plaintext $m$ to a given ciphertext $c$, i.e., $\forall m, c$ there exists exactly one value $k$ s.t. $E_k(m) = c$. So, the probability $\Pr[E_k(m) = c] = \Pr[k \text{ was drawn as key}] = 10^{-12}$ is independent of $m$.

## Exercise 5

What happens if the one-time pad is incorrectly used and that two distinct plaintexts are encrypted with the same key, and why?

1. The key is compromised.

2. The two plaintexts are revealed.

3. The difference between the two plaintexts is revealed.

4. The authenticity of the plaintext is compromised.

**Answer of exercise 5**

The difference between the two plaintexts is revealed.

In such a case, at each bit position, when the bits of the two plaintext are equal (resp. different), the ciphertext bits are equal (resp. different). Without knowing their individual values, the adversary can nevertheless tell where the two messages are equal and where they are different.

Mathematically, bitwise adding modulo-2 the two ciphertexts cancels the contribution of the key and reveals the bitwise difference between the two plaintexts (see the slides).

## Exercise 6

Suppose that the probability of wining the lottery is $\frac{1}{10^6}$. What is more likely guessing an AES-128 key on the first try or winning the lottery 6 times in a row? What about winning the lottery 7 times in a row?

**Answer of exercise 6**

Guessing an AES-128 key:

$$\frac{1}{2^{128}} = \frac{1}{2^8 \times 2^{120}} = \frac{1}{2^8 \times (2^{10})^{12}} = \frac{1}{256 \times (1024)^{12}}$$

Winning $n$ times in a row:

$$\left(\frac{1}{10^6}\right)^n = \frac{1}{(10^6)^n} = \frac{1}{((10^3)^2)^n} = \frac{1}{(1000)^{2n}}$$

## Exercise 7

Assume an adversary performs an exhaustive key search on a huge network of $10^9$ computers, each capable of testing $10^9$ keys per second. After about how much time will a 128-bit key typically be found?

1. A few seconds.
2. A few days.
3. A few years.

4. A few centuries.

5. A few times the age of the universe.

**Answer of exercise 7**

A few times the age of the universe.

There are $2^{128}$ keys to test, and on average the key will be found after scanning half the key space, so after testing $2^{128}/2 = 2^{127}$ keys. With this network, the adversary can test $10^{18}$ keys per second. So it will take $2^{127}/10^{18}$ seconds. Divide by 3600 to convert to hours, by 24 to days, by 365 to years, etc.

# Security definitions

## Exercise 8

Suppose that $(\text{Gen}, E, D)$ is an IND-CPA secure probabilistic encryption scheme, i.e., there is no known way of winning the IND-CPA game other than with a probability negligibly close to $\frac{1}{2}$ or with over-astronomical resources. Let the key and plaintext spaces be $\{0,1\}^n$ with $n \geq 128$. Point out the IND-CPA secure schemes in the following list:

- $E'_k(m) = E_k(m)||\text{first bit}(m)$
- $E'_k(m) = (\text{last bit}(m) \oplus \text{first bit}(m))||E_k(m)$
- $E'_k(m) = E_k(m)||1$
- $E'_k(m) = E_k(m)||E_k(m)$
  (+a side-question: will the first and last $n$ bits of $E'_k(m)$ always be equal?)
- $E'_k(m) = k||E_k(m)$
- $E'_k(m) = E_k(m)||(k \oplus 1^n)$
- $E'_{k1,k2}(m) = E_{k1}(m)||E_{k2}(m)$
- $E'_k(m) = E_{0^n}(m)$
- $E'_k(m) = E_k(m)||(m \oplus k)$

For each scheme that is *not* IND-CPA secure, please specify how an adversary can win the IND-CPA game, i.e., what the adversary chooses as plaintexts $m_0$ and $m_1$, what are the queries to be made before or after this choice, how the adversary distinguishes between the ciphertexts of $m_0$ and $m_1$.

**Answer of exercise 8**

- $E'_k(m) = E_k(m)||\text{first bit}(m)$: The adversary chooses $m_0$ and $m_1$ such that they differ in the first bit. It can recognize which plaintext was encrypted thanks to first bit$(m)$ exposed in the ciphertext. No need for queries during the game.

- $E'_k(m) = (\text{last bit}(m) \oplus \text{first bit}(m))||E_k(m)$: Same reasoning, except that the adversary chooses $m_0$ and $m_1$ such that they differ in either the first or the last bit (but not both).

+ $E'_k(m) = E_k(m)||1$: The extra bit 1 present in all ciphertexts do not help the adversary, so $E'$ is IND-CPA secure.

+ $E'_k(m) = E_k(m)||E_k(m)$: The two ciphertexts do not help the adversary, so $E'$ is IND-CPA secure. (The answer to the side-question is no: Not only the scheme is said to be probabilistic, if $(E, D)$ is IND-CPA secure, the adversary cannot even tell whether the same plaintext was encrypted twice. This means that the ciphertext space can be larger than the plaintext space, since one plaintext can be mapped to several ciphertexts.)

- $E'_k(m) = k||E_k(m)$: The key is revealed in the ciphertext, removing any security. To win the game, it suffices the adversary to use the revealed key to decrypt $E_k(m_0$ or $m_1)$ and find out which one it was.

- $E'_k(m) = E_k(m)||(k \oplus 1^n)$: Same reasoning, the key is also revealed since the mask $1^n$ is known.

+ $E'_{k1,k2}(m) = E_{k1}(m)||E_{k2}(m)$: The notation here suggests that the new encryption scheme has a $2n$-bit key, but each $n$-bit half is used independently. The adversary has twice more chance of winning the IND-CPA on $E'$ than on $E$, since distinguishing either $E_{k1}(m_0$ or $m_1)$ or $E_{k2}(m_0$ or $m_1)$ is sufficient. Twice a neglible probability is still a neglible probability, hence $E'$ is IND-CPA secure, although 1-bit less secure than $E$.

- $E'_k(m) = E_{0^n}(m)$: The scheme $E'$ does not make use of its secret key. Anything encrypted with $E_{0^n}$ can therefore be decrypted by the adversary with $D_{0^n}$, and it can immediately distinguish $E_{0^n}(m_0$ or $m_1)$ to win the game.

- $E'_k(m) = E_k(m)||(m \oplus k)$: As the plaintexts are chosen by the adversary, $m \oplus k$ reveals the key.

## Exercise 9

Let $(\text{Gen}, E, D)$ be a symmetric-key encryption scheme with diversification that is IND-CPA-secure. The definition of IND-CPA requires that the adversary respects the condition that the diversifier $d$ is a nonce. What would happen if this condition was not respected? Show how he can win this variant of IND-CPA game where $d$ is not unique, with the least number of queries.

**Answer of exercise 9**

The adversary chooses one plaintext $m_0$ and one diversifier value $d$ arbitrarily, and he asks for $c_0 = \text{Enc}_k(d, m_0)$. The adversary chooses another plaintext $m_1 \neq m_0$ but with the same length $|m_0| = |m_1|$, and submits $d$, $m_0$ and $m_1$ to the challenger. The challenger sends back $c = \text{Enc}_k(d, m_b)$ with unknown $b$. If $c = c_0$, then it means $m_0$ was encrypted and $b = 0$, otherwise it must be $m_1$ and $b = 1$.

The trick here is that the adversary uses the same value $d$ for his first query and for the encryption of the challenger. In the actual IND-CPA game, this is not allowed, and the encryption of $m_b$ by the challenger would be done with a different diversifier.

## Exercise 10

Suppose that $(\text{Gen}, E, D)$ is an IND-CPA secure symmetric-key encryption scheme with diversification, i.e., there is no known way of winning the IND-CPA game other than with a probability negligibly close to $\frac{1}{2}$ or with over-astronomical resources. Let the key space be $\{0,1\}^n$ with $n \geq 128$, the plaintext space to be arbitrary and the diversifier space be the set of non-negative integers $\mathbb{N}$. Point out the IND-CPA secure schemes in the following list:

- $E'_k(d, m) = E_k(0, m)$
- $E'_k(d, m) = E_k(d + 1, m)$
- $E'_k(d, m) = E_k(d, m)\|0^d$
- $E'_k(d, m) = E_k(\lfloor d/2 \rfloor, m)$
- $E'_k(d, m) = E_{0^d\|1^{n-d}}(d, m)$
- $E'_k(d, m) = E_k(\text{random} \in \{0, 1, \ldots, 36\}, m)$

- $E'_k(d, m) = E_k(\text{random} \in \{0, 1, \ldots, 2^{256}\}, m)$

For each scheme that is *not* IND-CPA secure, please specify how an adversary can win the IND-CPA game, i.e., what the adversary chooses as diversifiers, as plaintexts $m_0$ and $m_1$, what are the queries to be made before or after this choice, how the adversary distinguishes between the ciphertexts of $m_0$ and $m_1$.

**Answer of exercise 10**

- \- $E'_k(d, m) = E_k(0, m)$: Even if the adversary uses unique values for $d$ in $E'$, the diversifier is ignored and repeated internally when calling $E$. Hence, the encryption of identical plaintexts can be recognized. The adversary can submit for $m_0$ a previously queries plaintext and for $m_1$ an arbitrary one.

- \+ $E'_k(d, m) = E_k(d + 1, m)$: The diversifier is just invertibly remapped to a different value when calling $E$, so $E'$ is as secure as $E$.

- \+ $E'_k(d, m) = E_k(d, m) \| 0^d$: The value $d$ is public, so disclosing it in the ciphertext does not help the adversary.

- \- $E'_k(d, m) = E_k(\lfloor d/2 \rfloor, m)$: Both $d = 2n$ and $d = 2n + 1$ will be mapped to $n$ when using $E$. Hence, even if the adversary respects the unicity of $d$ in $E'$, the encryption of identical plaintexts can be recognized for pairs of queries. Concretely, the adversary can ask for one query with $d = 0$ and $m_0$, then submit $d = 1$, $m_0$ and an arbitrary $m_1$ to the challenger.

- \- $E'_k(d, m) = E_{0^d \| 1^{n-d}}(d, m)$: Here, $E$ is called with a publicly known key and the adversary can decrypt the ciphertext with $D_{0^d \| 1^{n-d}}$ to win the game.

- \- $E'_k(d, m) = E_k(\text{random} \in \{0, 1, \ldots, 36\}, m)$: In this case, $E$ is called with a random value instead of $d$. Given the small set of possible values, the adversary can expect that the diversifier used in $E$ will repeat. Concretely, the adversary can ask for the encryption of $m_0$ a hundred times or so, and with a high probability he will get all the possible ciphertexts corresponding to $m_0$. He then submits $m_0$ and an arbitrary $m_1$ and $d$. If $c$ is one of the possible ciphertexts for $m_0$, $b = 0$; otherwise, $b = 1$.

- \+ $E'_k(d, m) = E_k(\text{random} \in \{0, 1, \ldots, 2^{256}\}, m)$. This case is similar, but the cardinality of the random values used in $E$ is huge. Assuming a uniform distribution, the repetition of the random value is an event with a negligible probability and the adversary has only a negligible advantage over a random guess.

## Exercise 11

A bank issues a smartcard to each of its $N$ customers. Each smartcard $i$ contains its own secret key $K_i$ of $k$ bits. Because of the way the protocol was designed, each smartcard encrypts the string `"hello"` with its secret key and sends it over the network.

An attacker collects all these $N$ ciphertexts and starts a *multi-target exhaustive key search*. In more details, the attacker encrypts the string `"hello"` with each possible key in the key space until he gets a ciphertext that matches one of the collected ones. When it does, it probably means that he hits the secret key of one of the smartcards.

1. What is the security strength of the scheme under this attack as a function of $k$ and $N$?
2. If a bank issues a billion ($N \approx 2^{30}$) smartcards, each with its own $k = 128$-bit key, is the scheme still secure in practice?
3. What could the bank do to have a security strength of 128 bits?

### Answer of exercise 11

1. Each attempt has a chance $\frac{N}{2^k}$ to hit one of the $N$ keys. After $t$ attempts, the probabilty is $\epsilon(t) \approx \frac{tN}{2^k}$. We have

$$\log_2 t - \log_2 \epsilon(t) = \log_2 t - \log_2 \frac{tN}{2^k} = k - \log_2 N \geq s,$$

   hence the security strength is $k - \log_2 N$ in the absence of more efficient attacks. There is therefore a reduction of $\log_2 N$ bits of security compared to the nominal exhaustive key search.

2. If $N = 2^{30}$ and $k = 128$, the security strength is $128 - 30 = 98$ bits. It would take about $2^{98}$ attempts to hit one of the keys, which is still infeasible in practice today.

3. The bank could either increase the key length to 158 bits to compensate for the 30-bit loss, or fix its protocol so as to prevent multi-target exhaustive key search. For this latter case, the protocol could make the encryption depend on the smartcard identifier $i$ in some way (e.g., by including $i$ as associated data with authenticated encryption, see the appropriate chapter for more details). This forces the attacker to choose a value $i$, and therefore test only that $K_i$, at each attempt.

## Exercise 12

An automated teller machine (ATM) distributes cash to the customers of a bank. It is connected to the bank's server, which is in charge of authorizing and tracking transactions. The bank and the ATM share a $k$-bit secret key $K$. When authorizing, the bank sends the ATM a triplet $(u, m, \mathsf{tag})$ allowing user $u$ to receive an amount $m$ in cash, and $\mathsf{tag}$ is a $n$-bit message authentication code (MAC) computed under key $K$ over the first two components of the triplet. (This is of course a very simplified view for the sake of this exercise.)

1. What is the bank trying to protect against, assuming an adversary who has full access to the network connecting the bank and the ATM?

2. Let us assume for now that the bank and the ATM use a secure MAC function. What is the minimum key length $k$ that is required to have a security strength of 128 bits, and why?

3. Let us further assume that the ATM processes not more than one triplet per second and that the secret key $K$ is securely renewed every 24 hours. What is the minimum length $n$ of the MAC such that the probability of fraudulent transaction is less than $10^{-12}$ per day, and why?

### Answer of exercise 12

1. Without authentication, an attacker who has access to the network could insert messages that authorize fraudulent transactions. The MAC allows the ATM to distinguish fraudulent triplets from those legitimately created by the bank.

2. The minimum key size is 128 bits. With a smaller key size, exhaustive key search could be done with a time $t$ and a success probability $\epsilon$ that would violate the equation $\log_2 t - \log_2 \epsilon \geq s = 128$.

3. This is about random tag guessing. For a given key, the attacker can try to guess a tag $24 \times 3600 = 86400$ times; after that, the key is renewed and the attacker must start afresh. Correctly guessing a tag of $n$ bits succeeds with a probability $2^{-n}$ per attempt. To keep the probability of a correct key guess below $10^{-12}$ per day, we need to satisfy:

$$\begin{aligned} 86400 \times 2^{-n} &\leq 10^{-12} \\ 2^{-n} &\leq 10^{-12}/86400 \\ -n &\leq \log_2(10^{-12}/86400) \\ n &\geq \log_2(10^{12} \times 86400) \approx 56.3 \end{aligned}$$

Hence, the minimum tag size is 57 bits.

# Others

## Exercise 13

In order to save space or bandwidth we often use compression algorithms. Suppose you want to use data compression with encryption. How should these two operations be combined? Why?

**Answer of exercise 13**

Compress then encrypt. An encrypted message looks like random data and the compression algorithm will not be able to reduce the size.

## Exercise 14

Imagine that you have to use a relatively unreliable network i.e., packets often arrive with errors (bit flips) to their destination. Suppose you want to use an error correction code with encryption. How should these two operations be combined? Why?

**Answer of exercise 14**

Encrypt then apply error correction code. If the ciphertext is corrupted, the decryption algorithm will not work and there is no way to retrieve any meaningful information.

## Exercise 15

It is possible to use a symmetric crypto system to encrypt a short message e.g., 40-bit and obtain a 40-bit ciphertext. Could we create a secure public key crypto system with short ciphertexts?

**Answer of exercise 15**

No. Public key is known and one could encrypt all possible messages of short size.

## Exercise 16

A company that installs and maintains an open-source operating system is creating an automatic update mechanism for its customers. Each computer connected to the internet can fetch an *update pack* containing the latest changes to be made to the operating system. The company would like to guarantee the integrity of these

update packs so as to avoid the installation of malicious software on their customers' computers.

Please propose a solution based on schemes such as encryption, either public-key or secret-key and authentication, either MAC or signature. You also need to describe how the keys are distributed. Would a secret-key or a public-key approach be more appropriate?

**Answer of exercise 16**

As the operating system is open source, there is no confidentiality required, i.e., the update packs are distributed in the clear. However, their authenticity must be guaranteed, otherwise an adversary could install malicious software. Therefore, the update packs should come with a tag for the customers to be able to verify its authenticity.

If we go for a secret-key approach, this means that the authenticity is ensured with a message authentication code (MAC) computed over the update pack with a key secretely shared between a particular customer and the company. This means that the company must establish as many secret keys as it has customers. Perhaps the company can choose a secret key per customer and send it securely via a secure channel, but it is not so clear how.

If we go for a public-key approach, this means that the authenticity is sensured with a signature computed over the update pack with the company's private key, which can be verified with the company's public key. Here, a single private/public key pair is necessary. The company's public key can be bundled together with the first version of the operating system. Of course, the customer must be sure that the public key really belongs to the company, but this problem is not harder than ensuring that the first version of the operating system is genuine.

For the given reasons, the public key approach seems easier to manage: only one key instead of one per customer, and the distribution of the public key does not need confidentiality. Yet, the public-key approach still relies on the fact that the first version of the operating system and the company's public key arrive without corruption on the customer's computer.

## Exercise 17

Suppose Alice is broadcasting packets to 6 recipients $B_1, , B_6$. Privacy is not important but integrity is. In other words, each of $B_1, , B_6$ should be assured that the packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and $B_1, , B_6$ all share a secret key k. Alice computes a tag for every packet she sends using key $k$. Each user $B_i$ verifies the tag

when receiving the packet and drops the packet if the tag is invalid. Alice notices that this scheme is insecure because user $B_1$ can use the key $k$ to send packets with a valid tag to users $B_2, , B_6$ and they will all be fooled into thinking that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys $S = \{k_1, , k_4\}$. She gives each user $B_i$ some subset $S_i \subseteq S$ of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user Bi receives a packet he accepts it as valid only if all tags corresponding to his keys in $S_i$ are valid. For example, if user $B_1$ is given keys $\{k_1, k_2\}$ he will accept an incoming packet only if the first and second tags are valid. Note that $B_1$ cannot validate the 3rd and 4th tags because he does not have $k_3$ or $k_4$.

How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user? How many keys are needed for $t$ users if each of them receive only two keys?

**Answer of exercise 17**

$\{k_1, k_2\}, \{k_2, k_3\}, \{k_3, k_4\}, \{k_1, k_3\}, \{k_2, k_4\}, \{k_1, k_4\}$
Number of keys : $\lceil (1 + \sqrt{1 + 8t})/2 \rceil$