Exercises solutions of Cryptography
Q7 - LMAT2450

Benoît Legat Guillaume Gheysen Olivier Leblanc Luis Tascon Gutierrez

Jean-Martin Vlaeminck

Compilation: 06/01/2022 (11:57)

Last modification: 26/01/2020 (16:14) `7c6097bd`

Important Information This document is largely inspired from the excellent course given by François Koeune and Olivier Pereira at the EPL (École Polytechnique de Louvain), faculty of the UCL (Université Catholique de Louvain). It has been written by the aforementioned authors with the help of all other students, yours is therefore welcome as well. It is always possible to improve it, even more so if the course has changed in which case the exercises solutions must be updated accordingly. The source code and a link to the latest version of the pdf can be found at the following address

`https://github.com/Gp2mv3/Syntheses`.

There, you can also find the content of the `README` file which contains more information. You are invited to read it.

It is indicated there that questions, error reports, improvement suggestions or any discussion concerning the project are to be submitted at the following address

`https://github.com/Gp2mv3/Syntheses/issues`.

It allows everyone to see them, comment and act accordingly. You are invited to join the discussions.

You can also find informations on the wiki

`https://github.com/Gp2mv3/Syntheses/wiki`

like the status of the documents for each course

`https://github.com/Gp2mv3/Syntheses/wiki/Status`

You will have noticed that there are still a lot of missing ones, your help is welcome.

To contribute to the bug tracker or the wiki, you just have to create an account on GitHub. To interact with the source code of the documents, you will have to install LaTeX. To directly interact with the source code on GitHub, you will have to use `git`. If that constitutes a problem, we are of course open to contributions sent by mail (to contact.epldrive@gmail.com) or any other means.

Contents

Special thanks to the assistants Francesco Berti (francesco.berti@uclouvain.be) and Pierrick Méaux (pierrick.meaux@uclouvain.be) who gave us the LaTeX code of the statements of the exercises.

Note: I (Luis Tascon Gutierrez) had to merge the LaTeX code of the solution we have written and the code the assistants sent to me and it means that there might still be some typo errors due to commands that were not the same between the two documents.

Note: the distribution of the exercises changes from year to year. Here, the distribution of the year 2019–2020 is indicated.

APE 1

1.1: Exercise 1 (Vigenère)

You saw in the class the Vigenère encryption scheme. Write it formally as $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, with a key of length t, with $4 \leq t \leq 20$.

Solution to Exercise 1.1

- Gen: on input 1^t, pick $t \leftarrow \{4, \ldots, 20\}$, then pick $k \leftarrow \{0, \ldots, 25\}^t$ and output it as the key. Note that the usual n is here written t, as it is the length of the key.
- Enc: on input a key $k \in \{0, \ldots, 25\}^t$ and a message $m \in \{0, \ldots, 25\}^*$ of length $|m|$, construct the ciphertext

$$c = [(m_i + k_{i \mod t}) \mod 26]_{0 \leq i \leq |m|-1}$$

  This array-like notation means that we build each character of the ciphertext c_i by adding (modulo 26) the corresponding character of the plaintext m_i with the component $i \mod t$ of the key. All sequences (the message, the key and the ciphertext) are zero-indexed.

  A more "mathematical" approach:

  For $i = 1, \ldots, \lceil \frac{|m|}{t} \rceil$ do:

    For $j = 1, \ldots, t$ do:

      $c_{t(i-1)+j} = m_{t(i-1)+j} + k_j \mod 26$

  Return $c_1 || c_2 || \ldots || c_{|m|}$.

  This time, the sequences are 1-indexed.
- Dec: do the inverse: on input key $k \in \{0, \ldots, 25\}^t$ and message $m \in \{0, \ldots, 25\}^*$, construct the message

$$m = [(c_i - k_{i \mod t}) \mod 26]_{0 \leq i \leq |c|-1}$$

1.2: Exercise 2 (Perfect secrecy.)

We define the following encryption scheme for messages, keys and ciphertexts in $\mathbb{Z}_n$, where $\mathbb{Z}_n$ is essentially the integers in the interval $[0, n[$ (in fact $(\mathbb{Z}_n, +)$ forms a group):

- Gen outputs a key $k \in \mathcal{K}$ selected uniformly at random.
- $\mathsf{Enc}_k(m) \coloneqq k + m \mod n$
- $\mathsf{Dec}_k(c) \coloneqq c - k \mod n$

Suppose messages are drawn from $\mathcal{M}$ according to the binomial distribution. More precisely $M \sim \mathrm{Bi}(n-1, p)$ for some probability p which means that $\forall m \in \mathcal{M} : \Pr[M = m] = \binom{n-1}{m} p^m (1-p)^{n-1-m}$.

1. Show that the encryption scheme above is perfectly secret.

2. Evaluate $\Pr[C = c]$ for every $c \in \mathcal{C}$.
3. Evaluate $\Pr[K = k|C = c]$ for every $k \in \mathcal{K}$ and $c \in \mathcal{C}$.

Solution to Exercise 1.2

1. We have perfect secrecy if: $\Pr[C = c|M = m_0] = \Pr[C = c|M = m_1]$ for every $m_0, m_1 \in \mathcal{M}$ and $c \in \mathcal{C}$.

   Let $c \in \mathcal{C}$ and $m \in \mathcal{M}$. We have:

$$\begin{aligned}\Pr[C = c|M = m] &= \Pr[M + K = c \pmod n | M = m] \\ &= \Pr[m + K = c \pmod n] \\ &= \Pr[K = c - m \pmod n] \\ &= \frac{1}{n} \text{ (because } K \text{ is selected } \textbf{uniformly at random} \text{ in } \mathcal{K} \text{ where } |\mathcal{K}| = n) \\ &= \Pr[C = c|M = m'] \text{ for every } m' \in \mathcal{M}\end{aligned}$$

   Therefore, we have :

$$\Pr[C = c|M = m_1] = \Pr[C = c|M = m_2]$$

   for every $c \in \mathcal{C}$ and $m_1, m_2 \in \mathcal{M}$.

   Which means we have perfect secrecy.

2. Using the the result obtained at last exercice and the equivalent definitions about perfect secrecy, we can obtain:

$$\begin{aligned}\Pr[C = c] &= \Pr[C = c|M = m] \text{ for every } m \in \mathcal{M} \\ &= \frac{1}{n}\end{aligned}$$

   Other way to solve it (thanks to Benoît Legat):

$$\begin{aligned}\Pr[C = c] &= \sum_{m \in \mathcal{M}} \Pr[\mathsf{Enc}_K(M) = c|M = m] \Pr[M = m] \\ &= \frac{1}{n} \sum_{m \in \mathcal{M}} \Pr[M = m] \\ &= \frac{1}{n}.\end{aligned}$$

3.

$$\begin{aligned}\Pr[K = k|C = c] &= \Pr[C - M \equiv k \pmod n | C = c] \\ &= \Pr[c - M \equiv k \pmod n] \\ &= \Pr[M \equiv c - k \pmod n] \\ &= \binom{n-1}{c-k} p^{c-k} (1-p)^{n-1-c+k}.\end{aligned}$$

1.3: Exercise 4 (Negligible functions.)

1. Let f be a negligible function in n. Show that $g : n \mapsto 1000 \cdot f(n)$ is negligible too.
2. Show that the function $n \mapsto n^{-\log(n)}$ is negligible in n.

Solution to Exercise 1.3

1. Let p be a polynomial, let's take $q = 1000p$, since it is also a polynomial, we know that there exists N such that for all $n \geq N$,
$$f(n) \leq \frac{1}{q(n)}.$$
But that implies that
$$1000 \cdot f(n) \leq \frac{1}{p(n)}$$
so
$$g(n) \leq \frac{1}{p(n)}.$$

2. Let $p(n) = a_0 + a_1 n + \cdots + a_d n^d$ be an arbitrary polynomial of arbitrary degree d. Let N_1 such that $N_1 > r$ for each root r of p. We know that for $n \geq N_1$, the sign of p is the sign of a_d. Of course, if $a_d < 0$, our job is impossible but we do not consider these cases. Since $p(n) > 0$, our equation is equivalent to
$$n^{\log(n)} \geq p(n)$$
For $n \geq \max(N_1, 1)$ we also have $p(n) \leq n^d \sum_{i=0}^{d} |a_i|$. Taking the logarithm on both side (we can do it since the logarithm is strictly increasing), we have
$$\log^2(n) - d\log(n) - \log \sum_{i=0}^{d} |a_i| \geq 0$$
which is a second order polynomial in $\log(n)$. Let r_1, r_2 be its roots. We can take $N = \max(N_1, 1, 2^{r_1}, 2^{r_2})$.

There is another way to show this. We know that, f is negligible iff for all positive polynomial p, there exist an N such that for all $n \geq N$: $f(n) \leq \frac{1}{p(n)}$.

In our case we have $f(n) = n^{-log(n)}$ and we represent any polynomial as n^c. Then:
$$n^{-\log(n)} \leq n^{-c} \log(n^{-\log(n)}) \leq \log(n^{-c}) \log(n) \geq c$$
If we take $N = \exp(c)$, then our relation will be respected. As there exist an N where $n \leq N$ in wich the relation is respected, then the function is negligible.

1.4: Exercise 5 (Efficiency.)

Explain why the function that maps n on a sequence of "1" of length $\lfloor \sqrt{n} \rfloor$ cannot be evaluated by any efficient algorithm.

An example of such algorithm is given in Algorithm 1.

Require: $n \geq 0$
Ensure: A sequence of $\sqrt{n}$ "1"
  for $i = 0$ to $\lfloor \sqrt{n} \rfloor$ **do**
    Print '1'
  end for

Algorithm 1: example of algorithm

Hint: see n as a power of 2.

Solution to Exercise 1.4

An algorithm A is efficient if there exist a PPT p such that:

$$A(x) \leq p(|x|)$$

As we can see from the exercise:

$$A(n) \ = \ \sqrt{n}$$

$$|n| \ = \ \log_2(n) \textbf{ because n is encoded as a binary number}$$

(Here, the vertical bars don't represent the absolute value, but the "length" of the number.) But for all PPT p,

$$\sqrt{n} > p(\log_2(n))$$

So the algorithm is not efficient.

P.S.: It would have been efficient if we write the input as 1^n, because then the algorithm would have received an input of size n as an n-bit number.

Other more intuitive approach : The input n can be expressed under binary form as:

$$n = 2^{|n|}$$

Let's say that $k = |n|$. We know that the algorithm has to do at least $\sqrt{n}$ steps.

$$\sqrt{n} = \sqrt{2^k} = 2^{\frac{k}{2}}$$

Which is not polynomial.

APE 2

2.1: Exercise 1 (Security model.)

Let ϵ denote a negligible function. Remember that $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ has *indistinguishable multiple encryption in the presence of eavesdroppers* if $\forall$ PPT $\mathcal{A}$, $\exists$ ϵ:

$$\Pr[\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)\,,$$

where $\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$ is defined as follows.

1. $\mathcal{A}$ outputs $M_0 = (m_0^1, \ldots, m_0^t), M_1 = (m_1^1, \ldots, m_1^t)$
2. Choose $k \leftarrow \mathcal{G}(1^n)$ and $b \leftarrow \{0,1\}$, and send $(\mathsf{Enc}_k(m_b^1), \ldots, \mathsf{Enc}_k(m_b^t))$ to $\mathcal{A}$
3. $\mathcal{A}$ outputs b'
4. Define $\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) \coloneqq 1$ iff $b = b'$

Also remember that $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ has *indistinguishable encryption under a chosen-plaintext attack* if $\forall$ PPT $\mathcal{A}$, $\exists$ ϵ:

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)\,,$$

where $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ is defined as follows.

1. Choose $k \leftarrow \mathsf{Gen}(1^n)$
2. $\mathcal{A}$ **is given oracle access to** $\mathsf{Enc}_k(\cdot)$
3. $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}$
4. Choose $b \leftarrow \{0,1\}$ and send $\mathsf{Enc}_k(m_b)$ to $\mathcal{A}$
5. $\mathcal{A}$ **is again given oracle access to** $\mathsf{Enc}_k(\cdot)$
6. $\mathcal{A}$ outputs b'
7. Define $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) \coloneqq 1$ iff $b = b'$

Define the concept of indistinguishable *multiple* encryption under a chosen-plaintext attack.

Solution to Exercise 2.1

Two definition can be proposed. The first one is the one given in the reference [1, p. 84].

$\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ has indistinguishable *multiple* encryption under a chosen-plaintext attack if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ:

$$\Pr[\mathsf{PrivK}^{\mathsf{multcpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n),$$

where $\mathsf{PrivK}^{\mathsf{multcpa}}_{\mathcal{A},\Pi}(n)$ is defined as follows. This is the challenger's viewpoint.

1. Choose $k \leftarrow \mathsf{Gen}(1^n)$
2. $\mathcal{A}$ is given oracle access to $\mathsf{Enc}_k(\cdot)$
3. $\mathcal{A}$ outputs $M_0 = (m_0^1, \ldots, m_0^t)$, $M_1 = (m_1^1, \ldots, m_1^t)$
4. Choose $b \leftarrow \{0,1\}$, and send $(\mathsf{Enc}_k(m_b^1), \ldots, \mathsf{Enc}_k(m_b^t))$ to $\mathcal{A}$
5. $\mathcal{A}$ is again given oracle access to $\mathsf{Enc}_k(\cdot)$
6. $\mathcal{A}$ outputs b'

7. Define $\mathsf{PrivK}^{\mathsf{multcpa}}_{\mathcal{A},\Pi}(n) \coloneqq 1$ iff $b = b'$

An additional definition can be found in the reference book [1, p. 75]. Both are equally good since it can be proven they are equivalent to the definition of indistinguishably of a *single* encryption under CPA. Proving that, if Π has indistinguishable *multiple* encryption under CPA then it also has indistinguishable *single* encryption, is trivial. The other way is quite tricky. However in public key cryptosystems, CPA is the same as EAV since $\mathcal{A}$ has the public key and therefore oracle access to Enc_k. There is therefore the same property in asymmetric crypto for EAV than for symmetric crypto with CPA: equivalence of single- and multiple-message security. This is stated by [1, theorem 11.6] where it is proven. The proof is very similar to the proof needed to show the equivalence of single-message CPA and multiple-message CPA so if you are in doubt, just check it out.

2.2: Exercise 2 (Pseudorandomness.)

Let $F\colon \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^*$ be a (length-preserving) pseudorandom function, that is, if k is selected uniformly at random in $\{0,1\}^n$, then $F_k(\cdot)$ is computationnaly indistinguishable from a function f selected randomly in the set of functions from $\{0,1\}^n$ to $\{0,1\}^n$. More formally, $\forall$ PPT $\mathcal{D}$, $\exists$ negl. ϵ:

$$\left|\Pr[\mathcal{D}^{F_k(\cdot)}(1^n) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(1^n) = 1]\right| \leq \epsilon(n)$$

Show that F cannot seem random in front of an adversary who has an unbounded computational power, in the sense that she can distinguish it from a random function.

Solution to Exercise 2.2

There are $|\{0,1\}^n|^{|\{0,1\}|^n} = 2^{n2^n}$ function from $\{0,1\}^n$ to $\{0,1\}^n$. However, since there are only 2^n different k, F_k can only be 2^n different functions. If the distinguisher D^g is unbounded, he can just check the output of g for every possible input and for all $k \in \{0,1\}^n$, he can check if it has the same output of g. If it has the same output of F_k for at least one k, then $D^g(1^n) = 1$, else $D^g(1^n) = 0$. More formally

$$D^g(1^n) \triangleq \begin{cases} 1 & \text{if } \exists k \in \{0,1\}^n, \forall m \in \{0,1\}^n, F_k(m) = g(m) \\ 0 & \text{otherwise.} \end{cases}$$

If g is indeed a pseurorandom function, we can see that

$$\Pr[D^{F_k}(1^n) = 1] = 1$$

for all $k \in \{0,1\}^n$: we are guaranteed to detect it as we enumerate all possible F_k.

If g is a true random function, then the only case where we may be wrong is if the random function "mimics" one of the pseudorandom function, that is $\exists k\colon \forall m \in \{0,1\}^n\colon f(m) = F_k(m)$. Let's evaluate the probability that it happens:

$$|\{f\colon \{0,1\}^n \mapsto \{0,1\}^n \mid \exists k \in \{0,1\}^n, \forall m \in \{0,1\}^n f(m) = F_k(m)\}| \leq 2^n.$$

(the inequality is there to represent the fact that there could be k_1, k_2 such that $F_{k_1}(m) = F_{k_2}(m)$ for all $m \in \{0,1\}^n$.) Therefore

$$\Pr[D^f(1^n) = 1] \leq \frac{2^n}{2^{n2^n}} = \frac{1}{2^{n(2^n-1)}}.$$

Then, the difference between the probabilities is

$$\left|\Pr[D^{F_k}(1^n) = 1] - \Pr[D^f(1^n) = 1]\right| = 1 - \frac{1}{2^{n(2^n-1)}} \approx 1 \text{ for large values of } n$$

which is clearly non-negligible.

(Note that there is a slight chance that the distinguisher makes a mistake, but this probability decreases more than exponentially when n increases.)

2.3: Exercise 3 (Reduction.)

Let $\Pi := \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ be an encryption scheme having indistinguishable encryption under a chosen plaintext attack. Suppose we define a new scheme $\Pi' := \langle \mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}' \rangle$ as follows.

- $\mathsf{Gen}' := \mathsf{Gen}$
- $\mathsf{Enc}'_k(m) := \mathsf{Enc}_k(m)||1$ (i.e. a '1' bit is appended to the ciphertext)
- $\mathsf{Dec}'_k(c) := \mathsf{Dec}_k(c_1)$, where c_1 is obtained by discarding the last bit of c.

Is Π' also a CPA secure encryption scheme? Provide either an (efficient) attack/adversary or a (polynomial) reduction, depending on your claim.

Solution to Exercise 2.3

Π is a secure encryption scheme under CPA. Π is public, only the key is hidden from $\mathcal{A}$. Adding a 1 at the end will just give no information to $\mathcal{A}$.

This proof methodology is called "reduction".

Let $\mathcal{C}$ be the adversary trying to break Π and an efficient adversary $\mathcal{A}$ that can break Π' with a non-negligible probability. $\mathcal{O}$ is the oracle that gives $\mathcal{C}$ the challenge to break the scheme Π.

1. $\mathcal{O}$ is given 1^n as input as $\mathcal{C}$ that will transmit it to $\mathcal{A}$.
2. First query phase:
   - $\mathcal{A}$ outputs m_i as message to $\mathcal{C}$.
   - $\mathcal{C}$ outputs m_i as message to $\mathcal{O}$.
   - $\mathcal{O}$ outputs $c_i = Enc_k(m_i)$ as message to $\mathcal{C}$.
   - $\mathcal{C}$ sends back $c_i||1$ to $\mathcal{A}$.
3. Challenge phase:
   - $\mathcal{A}$ outputs m_0^*, m_1^* to $\mathcal{C}$.
   - $\mathcal{C}$ outputs m_0^*, m_1^* as message to $\mathcal{O}$.
   - $\mathcal{O}$ choose randomly $b \leftarrow \{0, 1\}$.
   - $\mathcal{O}$ outputs $c^* = Enc_k(m_b^*)$ to $\mathcal{C}$.
   - $\mathcal{C}$ sends back $c^*||1$ to $\mathcal{A}$.
4. Second query phase: same as the first one.
5. $\mathcal{A}$ outputs b' to $\mathcal{C}$.
6. $\mathcal{C}$ outputs b'.

We have:

$$\Pr[b' = b] = \Pr[\mathcal{A} \text{ wins over } \Pi']$$

If $\mathcal{A}$ has a non-negligible probability to win against the Π' scheme then $\mathcal{C}$ has also a non negligible probability to win against the Π scheme. We can conclude that Π' is also a secure scheme.

2.4: Exercise 4 (Reduction and/or attacks.)

Let $\Pi_1 = \langle \mathsf{Gen}^1, \mathsf{Enc}^1, \mathsf{Dec}^1 \rangle$ and $\Pi^2 = \langle \mathsf{Gen}^2, \mathsf{Enc}^2, \mathsf{Dec}^2 \rangle$ be an encryption scheme with $\mathsf{Enc}^1 \colon \mathcal{K} \times \mathcal{M}^1 \mapsto \mathcal{C}^1$ and $\mathsf{Enc}^2 \colon \mathcal{K} \times \mathcal{M}^2 \mapsto \mathcal{C}^2$

a. If $\mathcal{C}^1 = \mathcal{M}^2$, let $\Pi = \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ with

- $\mathsf{Gen} \coloneqq (\mathsf{Gen}_1, \mathsf{Gen}_2)$ (that is, we obtain two different keys (k_1, k_2)
- $\mathsf{Enc}_{(k_1,k_2)}(m) \coloneqq \mathsf{Enc}^2_{k_2}(\mathsf{Enc}^1_{k_1}(m))$
- $\mathsf{Dec}_{(k_1,k_2)}(c) \coloneqq \mathsf{Dec}^1_{k_1}(\mathsf{Dec}^2_{k_2}(c))$

1. If Π^1 is CPA secure, is it Π CPA secure?
2. If Π^2 is CPA secure, is it Π CPA secure?
3. If Π is CPA secure, is it Π^1 CPA secure?
4. If Π is CPA secure, is it Π^2 CPA secure?

b. If $\mathcal{M}^1 = \mathcal{M}^2$ and $\mathcal{C}^1 = \mathcal{C}^2$. let $\Pi' = \langle \mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}' \rangle$ with

- $\mathsf{Gen}' \coloneqq (\mathsf{Gen}^1, \mathsf{Gen}^2)$ (that is, we obtain two different keys (k_1, k_2)
- $\mathsf{Enc}'_{(k_1,k_2)}(m) \coloneqq (c_1, c_2)$ with $c_1 = \mathsf{Enc}^1_{k_1}(m),\ c_2 = \mathsf{Enc}^2_{k_2}(m))$
- $\mathsf{Dec}'_{(k_1,k_2)}(c) \coloneqq \mathsf{Dec}_{k_1}(c_1)$ with $c = c_1 \| c_2$ (c_1 is the first half of c)

5. If Π^1 is CPA secure, is it Π' CPA secure?
6. If Π^2 is CPA secure, is it Π' CPA secure?
7. If Π' is CPA secure, is it Π^1 CPA secure?
8. If Π' is CPA secure, is it Π^2 CPA secure?

Solution to Exercise 2.4

1. Let's assume Π is not CPA secure: There exist an adversary A.
   We build an adversary A^1 for Π_1

$$\Pr[b'' = b; b'' \leftarrow a^1] = \Pr[b' = b; b' \leftarrow a]$$
$$\Pr[b'' = b] = \Pr[b' = b] \leq 1/2 + \epsilon$$

   We will prove this by reduction. Let's assume we have a PPT adversary $\mathcal{A}_\Pi$ against Π with advantage ϵ. We build an adversary $\mathcal{A}^1 \coloneqq \mathcal{A}_{\Pi^1}$ against Π', running $\mathcal{A}_\Pi$ inside. $\mathcal{A}_{\Pi^1}$ has oracle access to $\mathcal{O}_{\Pi^1}$. We then run the following experiment (viewpoint of the adversary $\mathcal{A}_{\Pi^1}$):

   (1) The challenger-oracle chooses $k_1 \leftarrow \mathcal{K}$.

   (2) We, the adversary $\mathcal{A}_{\Pi^1}$, choose $k_2 \leftarrow \mathcal{K}$.

   (3) Query phase: for every request for encryption of message $m \in \mathcal{M}_1$ made by $\mathcal{A}_\Pi$ to an "oracle" $\mathcal{O}_\Pi$ (played by us), we redirect this request to our oracle $\mathcal{O}_{\Pi^1}$ and receive $c = \mathsf{Enc}^1_{k_1}(m)$. We then compute $c' = \mathsf{Enc}^2_{k_2}(c) = \mathsf{Enc}^2_{k_2}(\mathsf{Enc}^1_{k_1}(m)) = \mathsf{Enc}_{(k_1,k_2)}(m)$ and forward it to the "emulated" adversary $\mathcal{A}_\Pi$.

   (4) Challenge phase: the adversary $\mathcal{A}_\Pi$ outputs two messages m_0 and $m_1 \in \mathcal{M}^1$, we output both to the challenger-oracle.

   (5) The challenger-oracle chooses $b \leftarrow \{0, 1\}$ and sends $c^* = \mathsf{Enc}^1_{k_1}(m_b)$ to us. We compute $c^{*'} = \mathsf{Enc}^2_{k_2}(c^*) = \mathsf{Enc}_{(k_1,k_2)}(m_b)$ and send it to $\mathcal{A}_\Pi$.

   (6) Second query phase, similar to the first one.

   (7) $\mathcal{A}_\Pi$ outputs its guess b'. We output $b'' \coloneqq b'$.

From the point of view of $\mathcal{A}_\Pi$, it has discuted with an oracle and challenger for Π: it has thus executed in the correct conditions and so, its advantage is ϵ. In addition, our constructed adversary $\mathcal{A}_{\Pi^1}$ has only executed PPT algorithms in addition to $\mathcal{A}_\Pi$, so the overall adversary is PPT. We can then compute the probability of success of our adversary $\mathcal{A}_{\Pi^1}$:

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\Pi^1},\Pi^1}(n)=1] = \Pr[b''=b] = \Pr[b'=b] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1] = \frac{1}{2}+\epsilon(n),$$

the same probability as $\mathcal{A}_\Pi$. As we know that Π^1 is CPA-secure, then it means that $\epsilon(n)$ is negligible, and so Π is also CPA-secure.

2. The proof is similar to the above, but we send $m' = \mathsf{Enc}^1_{k_1}(m)$ to our oracle for $\mathcal{O}_{\Pi^2}$ and directly get $c = \mathsf{Enc}^2_{k_2}(\mathsf{Enc}^1_{k_1}(m)) = \mathsf{Enc}_{(k_1,k_2)}(m)$.

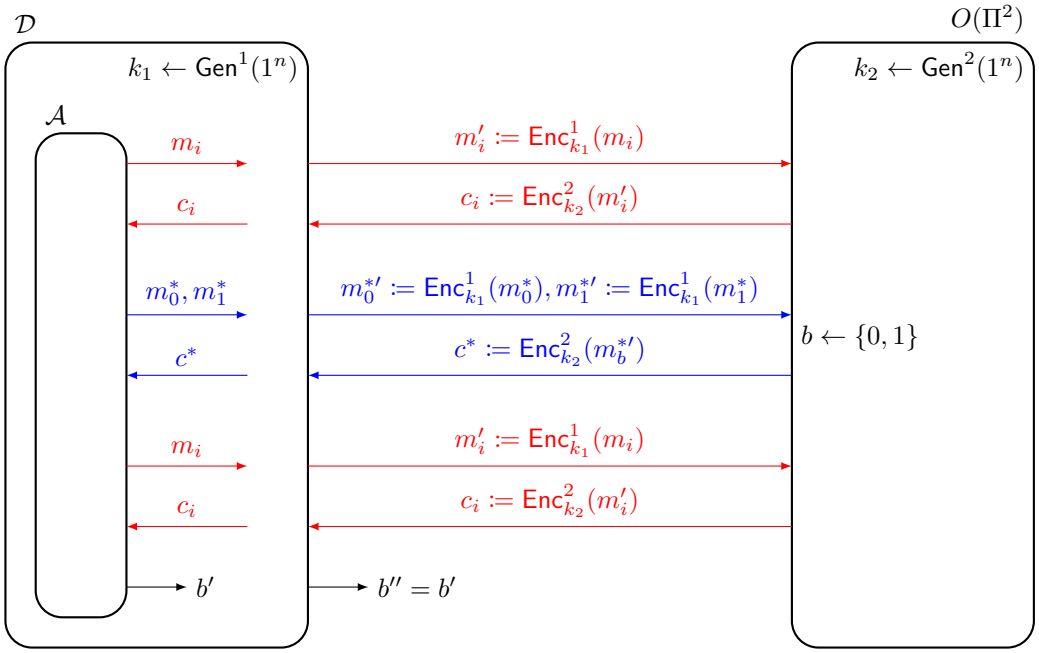
   If Π^2 is CPA secure, is it Π CPA secure?

   We ($\mathcal{D}$) define an oracle ($O(\Pi^2)$) that can securely encode a message with Π^2 and instantiate an Attacker ($\mathcal{A}$). As we have to challenge the Π scheme knowing the Π^2 is CPA secure we will proceed as follow.

   First learning phase: We begin by encrypting the messages from the attacker with Π^1 to send them to the oracle. The oracle responds by encrypting the message received with Π^2 and we just pass this response to the attacker.

   Challenge phase: The attacker choose two messages and we transmit the two messages with the first encryption. The oracle will choose witch message to encrypt and will respond with one of the two messages encrypted that we will send back to the attacker.

   Second learning phase: Same as the first one.



As we can see in every case, the distinguisher will have the same probability to find the message encrypted by the oracle than the attacker to break the scheme. As the attacker can only have a probability of $1/2+\epsilon$ to succeed the distinguisher will have the same probability. So, the scheme Π is secure.

3. As seen in the previous development, if Π^2 is CPA secure, Π is CPA secure. There is no restriction on Π^1 in that case. Therefore Π^1 could be such that $\mathsf{Enc}^1_{k_1}(m) \coloneqq m$ which is obviously not CPA secure. So the proposition is false.

4. Idem

5. The Π' scheme is CPA secure if and only if Π^2 is also CPA secure. For example, if $\mathsf{Enc}^2_{k_2}(m) = m$ then the scheme Π' is obviously not CPA secure.

   More formally: Let's define the following experiment (the viewpoint is that of the adversary $\mathcal{A}$):

   (1) The challenger chooses $k_1, k_2 \leftarrow \mathcal{K} \times \mathcal{K}$.

   (2) No query phase takes place

   (3) We output $m_0 = 0^{|\mathcal{M}_1|}$ and $m_1 = 1^{|\mathcal{M}_2|}$ to the challenger.

   (4) The challenger chooses $b \leftarrow \{0,1\}$ and returns $c = (\mathsf{Enc}^1_{k_1}(m_b), m_b)$ (recall that $\mathsf{Enc}^2_{k_2}(\cdot)$ is the identity here).

   (5) We output $b' =$ last bit of c.

   Then it is clear that $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi'}(1^n) = 1] = \Pr[b' = b] = 1$.

6. Similar to the previous one: assume Enc^1 is the identity.

7. We will proof this by reduction. Again, let's suppose we have a PPT adversary $\mathcal{A}_{\Pi^1}$ for Π^1 with advantage ϵ, and let's build another PPT adversary $\mathcal{A}_{\Pi'}$ against Π'. The adversary $\mathcal{A}_{\Pi'}$ has access to an oracle $\mathcal{O}_{\Pi'}$ for Π', and runs the adversary $\mathcal{A}_{\Pi^1}$ inside. We then run the following experiment (viewpoint of $\mathcal{A}_{\Pi'}$):

   (1) $\mathcal{A}_{\Pi^1}$ chooses the security parameter n, sends it to $\mathcal{A}_{\Pi'}$, which sends it to the oracle. *(Note: I'm not entirely sure of this one.)*

   (2) The oracle chooses $k_1, k_2 \leftarrow \mathcal{K} \times \mathcal{K}$.

   (3) First query phase: When $\mathcal{A}_{\Pi^1}$ sends a query for encryption of message m to its "oracle" $\mathcal{O}_{\Pi^1}$ (played by us), we forward it to our oracle $\mathcal{O}_{\Pi'}$, which sends us back an encryption $c = \mathsf{Enc}'_{(k_1,k_2)}(m) = (c_1, c_2) = (\mathsf{Enc}^1_{k_1}(m), \mathsf{Enc}^2_{k_2}(m))$. We forward c_1 to $\mathcal{A}_{\Pi^1}$.

   (4) Challenge phase: $\mathcal{A}_{\Pi^1}$ outputs two messages $m_0, m_1 \in \mathcal{M}^1$. We output them to the challenger-oracle.

   (5) The challenger-oracle chooses $b \leftarrow \{0,1\}$ and returns $c^* = \mathsf{Enc}'_{(k_1,k_2)}(m_b) = (c_1^*, c_2^*) = (\mathsf{Enc}^1_{k_1}(m_b), \mathsf{Enc}^2_{k_2}(m_b))$ with c_1 and c_2 of equal length.

   (6) We return c_1^* to $\mathcal{A}_{\Pi^1}$ as the challenge of the "oracle" $\mathcal{O}_{\Pi^1}$.

   (7) Second query phase, similar to the first one.

   (8) $\mathcal{A}_{\Pi^1}$ outputs its guess b'. We output $b'' \coloneqq b'$ to the challenger.

   From the point of view of $\mathcal{A}_{\Pi^1}$, it has communicated with an oracle for Π^1, and so it has executed in the correct conditions and its advantage in guessing b is ϵ. As our adversary $\mathcal{A}_{\Pi'}$ only executes PPT algorithms, it is PPT too. We can compute the probability of success of $\mathcal{A}_{\Pi'}$:

   $$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\Pi'},\Pi'}(n) = 1] = \Pr[b'' = b] = \Pr[b' = b] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\Pi^1},\Pi^1}(n) = 1] = \frac{1}{2} + \epsilon$$

   As we know that Π' is CPA-secure, then ϵ must be negligible, and so Π^1 is also CPA-secure.

   Intuition: Π' effectively executes the two encryption schemes in parallel, with no communication between the two. If one of the two encryption scheme was not CPA-secure, then we could just focus on this scheme to break the whole Π'.

8. Similar to the previous one; the only difference is that we send c_2 instead of c_1.

APE 3

3.1: Exercise 0 (Simple attacks)

Let $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be existentially unforgeable under an adaptive chosen-message attack and let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure scheme. Consider the following schemes $\mathsf{MAC}' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ (resp. $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$) based on Mac with $\mathsf{Gen}' = \mathsf{Gen}$ and Mac' (resp. $\mathsf{Gen} = \mathsf{Gen}'$ and Enc') defined as follow:

1. $\mathsf{Mac}'_k(m) \coloneqq (\mathsf{Mac}_k(m), \mathsf{Mac}_k(m \oplus 0 \dots 01))$
2. $\mathsf{Mac}'_k(m) \coloneqq \mathsf{Mac}_k\left(\bigoplus_{i=1}^{l} m_i\right)$
3. $\mathsf{Mac}'_k(m) \coloneqq (\mathsf{Mac}_k(m_1), \dots, \mathsf{Mac}_k(m_l))$
4. $\mathsf{Mac}'_k(m) \coloneqq (\mathsf{Mac}_k(m_1), \mathsf{Mac}_k(m_1||m_2), \dots, \mathsf{Mac}_k(m_1|| \dots ||m_l))$
5. $\mathsf{Enc}'_k(m) \coloneqq \left(\mathsf{Enc}_k(m), \mathsf{Enc}_k\left(\bigoplus_{i=1}^{l} m_i\right)\right)$
6. $\mathsf{Enc}'_k(m) \coloneqq \left(\mathsf{Enc}_k(m), \bigoplus_{i=1}^{l} m_i\right)$
7. $\mathsf{Enc}'_k(m) \coloneqq (\mathsf{Enc}_k(m), \mathsf{Enc}_k(m \oplus 110 \dots 0))$
8. $\mathsf{Enc}'_k(m) \coloneqq (\mathsf{Enc}_k(m||0), \mathsf{Enc}_k(m))$

Break all MAC' and Π'.

(In some cases m is parsed in $m_1, \dots, m_l$ with $|m_1| = \dots = |m_{l-1}| = n$, $|m_l| \leq n$ and $m_1|| \dots ||m_l = m$ ($||$ is the concatenation) where n is the security parameter.)

Solution to Exercise 3.1

The following are sketches of the attacks that need to be performed; a correct answer needs to define the setting of the attack, in a manner similar to the reduction proofs.

1. Let's build an adversary $\mathcal{A}$ that can break the MAC' scheme. Let's define the following experiment $\mathsf{MacForge}_{\mathcal{A},\Pi'}(n)$ (in the viewpoint of the adversary):

   (1) The oracle-challenger chooses $k \leftarrow \mathcal{K}$.

   (2) We are given access to an oracle for $\mathsf{Mac}'_k(\cdot)$; this oracle records all requested messages in Q. With $m \coloneqq 0 \dots 0$, we ask for $t \coloneqq \mathsf{Mac}'_k(m)$ and receive $t = (t_1, t_2)$ with $t_1 = \mathsf{Mac}_k(m)$ and $t_2 = \mathsf{Mac}_k(m \oplus 0 \dots 01)$.

   (3) We output $m^* = 0 \dots 01$ and $t^* = (t_2, t_1)$.

   (4) Define $\mathsf{MacForge}_{\mathcal{A},\Pi'} \coloneqq 1$ iff $\mathsf{Vrfy}'_k(m^*, t^*) = 1$.

   Observe that

   $$\begin{aligned}\mathsf{Mac}'_k(m^*) &= (\mathsf{Mac}_k(m^*), \mathsf{Mac}_k(m^* \oplus 0 \dots 01)) = (\mathsf{Mac}_k(0 \dots 01), \mathsf{Mac}_k(0 \dots 01 \oplus 0 \dots 01)) \\ &= (\mathsf{Mac}_k(0 \dots 01), \mathsf{Mac}_k(0 \dots 0)) = (t_2, t_1)\end{aligned}$$

   and so, the pair (m^*, t^*) is indeed a valid tag. In addition, the message m^* is different from m and so $m^* \notin Q$. Thus, $\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi'} = 1] = 1$ which is clearly not negligible.

2. We ask $m = 0 \dots 0||0 \dots 01$ to the oracle as a message consisting of two blocks of length l, and receive the tag $t = \mathsf{Mac}'_k(m) = \mathsf{Mac}_k(0 \dots 0 \oplus 0 \dots 01) = \mathsf{Mac}_k(0 \dots 01)$.

Then, we output $m^* \coloneqq 0\ldots01||0\ldots0$ and $t^* \coloneqq t$. Observe that $\mathsf{Mac}'_k(m^*) = \mathsf{Mac}_k(0\ldots01 \oplus 0\ldots0) = \mathsf{Mac}_k(0\ldots01) = t = t^*$ and thus is a valid tag for a message not asked to the oracle.

3. We send to the oracle $m = 0\ldots0||0\ldots01$ as a two-blocks message and receive $t = \mathsf{Mac}'_k(m) = (\mathsf{Mac}_k(0\ldots0), \mathsf{Mac}_k(0\ldots01)) = (t_1, t_2)$. We output $m^* = 0\ldots01||0\ldots0$ and $t^* = (t_2, t_1)$. Observe that $\mathsf{Mac}'_k(m^*) = (\mathsf{Mac}_k(0\ldots01), \mathsf{Mac}_k(0\ldots0)) = (t_2, t_1) = t^*$, and m^* was not asked to the oracle, so this is a valid pair.

4. We send $m = m_1||m_2$ for some message blocks $m_1, m_2 \leftarrow \{0,1\}^n$ (they don't matter) and receive $t = \mathsf{Mac}'_k(m) = (\mathsf{Mac}_k(m_1), \mathsf{Mac}_k(m_1||m_2)) = (t_1, t_2)$. We output $m^* = m_1$ and $t^* = (t_1)$. Note that this requires $\mathsf{Mac}_k(\cdot)$ and $\mathsf{Mac}'_k(\cdot)$ to accept arbitrary-length messages, and $\mathsf{Mac}'_k(\cdot)$ to output arbitrary-length tags.

5. We ask the oracle to encrypt $m = 0\ldots01||0\ldots01$, a two-blocks message and receive $c = (c_1, c_2) = (\mathsf{Enc}_k(m), \mathsf{Enc}_k(0\ldots01 \oplus 0\ldots01)) = (\mathsf{Enc}_k(m), \mathsf{Enc}_k(0\ldots0))$. We then output $m_0 = 0\ldots0||0\ldots0$ and $m_1 = 1\ldots1||1\ldots1$. Observe that $\bigoplus_{i=1}^{l} m_{0,i} = \bigoplus_{i=1}^{l} m_{1,i} = \bigoplus_{i=1}^{l} m_i = 0\ldots0$. We receive $c = (c_1^*, c_2^*) = (\mathsf{Enc}_k(m_b), \mathsf{Enc}_k(\bigoplus_{i=1}^{l} m_{b,i})) = (\mathsf{Enc}_k(m_b), \mathsf{Enc}_k(00\ldots00))$. We then ask to decrypt $c^* = (c_1^*, c_2)$ and will receive m_b.

   This assumes that the two encryptions that take place in Enc' are independent (e.g., use different r) so that we can combine the two parts in arbitrary ways. Also, there is a slight chance that $c_2 = c_2^*$ (if both encryptions used the same random values) as they encrypt the same message. If n is the number of bits of the random values used in the encryption (e.g., the number of bits of r), then this probability is $\frac{1}{2^n}$ which is negligible.

6. We output $m_0 = 0\ldots0$ and $m_1 = 0\ldots01$ with length n (so that the messages are not split). Then, receiving $c = (\mathsf{Enc}_k(m_b), \bigoplus_{i=1} lm_b) = (\mathsf{Enc}_k(m_b), m_b)$, we answer 0 if $m_b = m_0$ and 1 otherwise.

7. We output $m_0 = 0\ldots0$ and $m_1 = 1\ldots1$, and receive $c = (\mathsf{Enc}_k(m_b), \mathsf{Enc}_k(m_b \oplus 110\ldots0)) = (c_1, c_2)$. We then ask the oracle to decrypt $c^* = (c_2, c_1)$ (this is not c) and receive $m^* = m_b \oplus 110\ldots0$. We then just have to compute $m' = m^* \oplus 110\ldots0$ and compare with m_0 and m_1 to identify it. For this attack to work, we need a decryption oracle, and so need to play the CCA security game.

8. We output $m_0 = 0\ldots0$ and $m_1 = 1\ldots1$ and receive $c = (\mathsf{Enc}_k(m_b||0), \mathsf{Enc}_k(m_b)) = (c_1, c_2)$. We ask the oracle to decrypt $c^* = (c_2, c_1)$ and should receive $m_b||0$ or m_b.

   Note: this attack looks wrong, as it requires Dec' to ignore the fact that, in theory, the first part of the ciphertext encrypts the same message as the second part, with a 0 appended. In practice, decrypting such malformed messages should result in an error. But it was the "official" answer.

3.2: Exercise 1 (Fixed-length MAC)

Consider the fixed-length MAC $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy} \rangle$ defined as follows:

- Gen: choose random $k \leftarrow \{0,1\}^n$
- Mac: on input $m, k \in \{0,1\}^n$, output $t \coloneqq F_k(m)$
- Vrfy: on input $k, m, t \in \{0,1\}^n$ output 1 iff $t = F_k(m)$

Prove that, if F is a PRF, Π is existentially unforgeable under an adaptive chosen-message attack. Hint:

1. Consider the scheme Π' defined as Π except that a truly random function is used instead of a pseudo-random one. Show that Π' is existentially unforgeable under an adaptive chosen-message attack.

2. Consider a PPT adversary who can produce an adaptive forgery on Π with non negligible probability $\epsilon(n)$. Using this adversary, show that F cannot be a PRF.

Solution to Exercise 3.2

- Let $\tilde{\Pi} = \langle \tilde{\mathsf{Gen}}, \tilde{\mathsf{Mac}}, \tilde{\mathsf{Vrfy}} \rangle$, defined as:
  - $\tilde{\mathsf{Gen}}$: chooses a random f.
  - $\tilde{\mathsf{Mac}}$: on input m, outputs $f(m)$.
  - $\tilde{\mathsf{Vrfy}}$: on input (m, t), outputs 1 iff $f(m) = t$.

  Let's analyse the maximum value of $\Pr[\mathsf{MacForge}_{\mathcal{A},\tilde{\Pi}}(n) = 1]$ for an adversary $\mathcal{A}$. If after q different queries (it gains no info doing the same query twice), $m_1, \ldots, m_q$, $\mathcal{A}$ outputs (m, t), what are its chances of success ? Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$. There are $(2^n)^{2^n}$ different f and we pick a random one uniformly. However, there are only $(2^n)^{2^n-q}$ experiments such that $\mathcal{A}$ could have received (m_i, t_i) for $i = 1, \ldots, q$ because there are $(2^n)^{2^n-q}$ f such that $f(m_i) = t_i$ for $i = 1, \ldots, q$. We could be in any of them. Among them, only $(2^n)^{2^n-(q+1)}$ are such that $f(m) = t$. Since f is selected uniformly, we have

$$\begin{aligned}
\Pr[\mathsf{MacForge}_{\mathcal{A},\tilde{\Pi}}(n) = 1] &= \Pr[f(m) = t | f(m_i) = t_i, \forall i = 1, \ldots, q] \\
&= \frac{\Pr[f(m) = t, f(m_i) = t_i, \forall i = 1, \ldots, q]}{\Pr[f(m_i) = t_i, \forall i = 1, \ldots, q]} \\
&= \frac{\frac{(2^n)^{2^n-(q+1)}}{(2^n)^{2^n}}}{\frac{(2^n)^{2^n-q}}{(2^n)^{2^n}}} = \frac{(2^n)^{2^n-(q+1)}}{(2^n)^{2^n-q}} = \frac{1}{2^n}.
\end{aligned}$$

  A shortcut would have been to say that, since $f(m)$ is independent of the $f(m_i)$, we have

$$\begin{aligned}
\Pr[\mathsf{MacForge}_{\mathcal{A},\tilde{\Pi}}(n) = 1] &= \Pr[f(m) = t | f(m_i) = t_i, \forall i = 1, \ldots, q] \\
&= \Pr[f(m) = t] = \frac{(2^n)^{2^n-1}}{(2^n)^{2^n}} = \frac{1}{2^n}.
\end{aligned}$$

  Another way of saying it: as f is random, each $f(m_i)$ is random too and independent of each other and of m, so $\mathcal{A}$ doesn't get any information from the query phase. And, because f is random, the correct tag t^* for m is random too, and so the adversary only has $\Pr[t = t^*] = \frac{1}{2^n}$ to find the correct one.

  It is quite surprising that instead of a upper bound on $\Pr[\mathsf{MacForge}_{\mathcal{A},\tilde{\Pi}}(n) = 1]$ depending on $\mathcal{A}$ (and reached for $\mathcal{A}$ super smart), it is actually independent of $\mathcal{A}$.

- Let's now suppose that we have an adversary $\mathcal{A}$ that win with non-negligible probability against the MAC Π and show that under this assumption we can build a distinguisher $\mathcal{D}$ for F.

  $\mathcal{D}$ will simply take a function g as input and run $\mathcal{A}$ using g to create the tags. He has g so he can see if $\mathcal{A}$ wins or lose. If $\mathcal{A}$ wins, $\mathcal{D}$ outputs 1, otherwise, it outputs 0.

  We know that if g is a pseudo random function, $\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi_g} = 1] = \frac{1}{2^n}$ (we prove that in the previous point) and if g is a PRF $\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi_g} = 1] = \eta(n)$ where η is non-negligible ($\mathcal{A}$ is exactly in the correct conditions he expects to work correctly and to win with non-negligible probability). We have therefore

$$|\Pr[\mathcal{D}^{F_k}(1^n) = 1] - \Pr[\mathcal{D}^f(1^n) = 1]| = \left|\eta(n) - \frac{1}{2^n}\right| \geq \eta(n) - \frac{1}{2^n}$$

which is non-negligible due to $\eta(n)$. Conversely, as we assume that F is a PRF, then it must be that the probability on the left-hand side is negligible, and so the probability on the right-hand side must be negligible too, which forces $\eta(n)$ to be negligible.

3.3: Exercise 2 (MAC length extension)

Let $\Pi' := \langle \mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}' \rangle$ be a secure fixed-length MAC. We define a variable-length MAC $\Pi := \langle \mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy} \rangle$ as follows:

- Gen: choose random $k \leftarrow \{0,1\}^n$
- Mac: on input $k \in \{0,1\}^n$ and $m \in \{0,1\}^*$ of length $l < 2^{\frac{n}{4}}$
  - Parse m into blocks $m_1, \ldots, m_d$ of length $\frac{n}{4}$ each (pad with 0's if necessary)
  - Choose random $r \leftarrow \{0,1\}^{\frac{n}{4}}$
  - Compute $t_i := \mathsf{Mac}_k(r||l||i||m_i)$ for $1 \leq i \leq d$, with $|r| = |l| = |i| = \frac{n}{4}$
  - Output $t := \langle r, t_1, \ldots, t_d \rangle$
- Vrfy: on input $k, m, t = \langle r, t_1, \ldots, t_{d'} \rangle$,
  - Parse m into blocks $m_1, \ldots, m_d$ of length $\frac{n}{4}$ each
  - Output 1 iff $d = d'$ and, $\forall 1 \leq i \leq d$, $\mathsf{Vrfy}'_k(r||l||i||m_i, t_i) = 1$

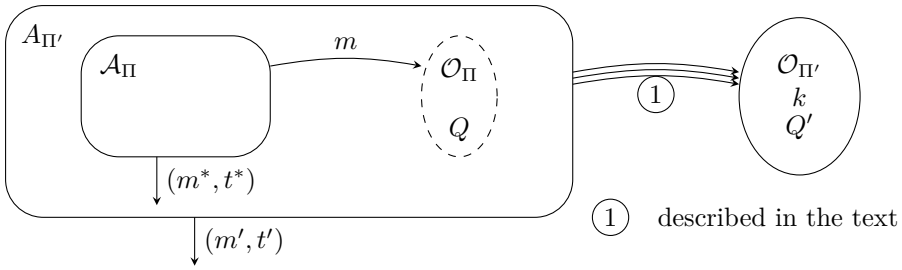
The goal of this exercise is to prove by reduction that Π is existentially unforgeable. Let $\mathcal{A}$ (resp. $\mathcal{A}'$) be an adversary attacking the unforgeability of Π (resp. Π') and let $\epsilon = \textsc{MacForge}_{\mathcal{A},\Pi}(n)$ (resp. $\epsilon' = \textsc{MacForge}_{\mathcal{A}',\Pi'}(n)$) denotes its advantage.

1. Make a quick draw sketching the proof.
2. Describe formally how $\mathcal{A}'$ should react to a query $\textsc{Mac}_k(m)$.
3. Define what is a mac forgery for the scheme Π. Does it necessarily implie a forgery on the scheme Π' (justify your answer).
4. Express ϵ in function of ϵ' and conclude.

Solution to Exercise 3.3

(See [1, pp. 120–122] for the "official" proof).

1.



2. $\mathcal{A}'$ should pick a random $r \leftarrow \{0,1\}^{\frac{n}{4}}$, parse m into d blocks $m_1, ..., m_d$ and send to its oracle d queries $r\|l\|i\|m_i$ for $i = 1, \ldots, d$. After the d queries made to its oracle O', $\mathcal{A}'$ computes $\langle r, t_1, \ldots, t_d \rangle$ where t_i is the answer of the oracle to its ith query. He then sends it as an answer to $\mathcal{A}$s query.
3. For Π, a forgery is a pair $(m, \langle r, t_1, \ldots, t_d \rangle)$ where m has not been queried before and $t_i = \mathsf{Mac}'_k(r\|l\|i\|m_i)$ for $i = 1, \ldots, d$ where l is the length of m and $m = m_1\| \cdots \|m_d$.
   - If none of its previous query has the same r and l, $r\|l\|1\|m_1$ cannot be a query made by

$\mathcal{A}'$ and $(r\|l\|1\|m_1, t_1)$ is an existential forgery for Π' that $\mathcal{A}'$ can output.

- If 2 previous queries have both r and l, then we do not necessarily have an existential forgery. However, since r is picked at random ($\mathcal{A}'$ could cheat and make sure that the same r is not picked twice) the probability (birthday paradox) of this to happen (if all m have the same l which is the worst case) is approximately $\frac{q(n)^2}{2 \cdot 2^n}$ where $q(n)$ is the number of queries made by $\mathcal{A}$.
- If one unique previous query m^j has this r and l, since m is not one of the previous query, there must be i such that $m_i \neq m_i^j$. We know therefore that $r\|l\|j\|m_j$ has never been queried by $\mathcal{A}'$ so $(r\|l\|j\|m_j, t_j)$ is an existential forgery for Π.

4. In conclusion, we have

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \Pr[\mathsf{MacForge}_{\mathcal{A}',\Pi'}(n) = 1] + \frac{q(n)}{2^{n+1}}$$
$$\epsilon(n) \leq \epsilon'(n) + \frac{q(n)}{2^{n+1}}$$

so since $\epsilon'(n)$ and $\frac{q(n)}{2^{n+1}}$ are negligible, $\epsilon(n)$ is negligible.

Alternative answer:

2. When $\mathcal{A}_\Pi$ asks for $\mathsf{Mac}_k(m)$, we ($\mathcal{A}_{\Pi'}$) should parse $m = m_1||m_2||\dots||m_d$ with $|m_i| = \frac{n}{4} \, \forall 1 \leq i \leq d = \lceil \frac{l}{n/4} \rceil$ and with the message zero-padded.

   Pick $r \leftarrow \{0,1\}^{n/4}$.

   Then, do d requests to $\mathcal{O}_{\Pi'}$ to build $t_i \coloneqq \mathsf{Mac}'_k(r||l||i||m_i) \, \forall 1 \leq i \leq d$.

   Finally, send $t = \langle r, t_1, \dots, t_d \rangle$ to $\mathcal{A}_\Pi$.

   An important question to answer is whether this operation keeps $\mathcal{A}_{\Pi'}$ PPT, with so much requests. To prove this, observe that 1) the message m is generated by a PPT adversary so 2) its length must be polynomial and 3) there is thus at most a polynomial number of blocks and thus 4) we only do a polynomial number of requests to $\mathcal{O}_{\Pi'}$ when $\mathcal{A}_\Pi$ does a request to $\mathcal{O}_\Pi$. So, we are still PPT.

3. When $\mathcal{A}_\Pi$ outputs (m^*, t^*), we have $t^* = \langle r, t_1, \dots, t_d \rangle$ with $t_i = \mathsf{Mac}'_k(r||l||i|m_i^*)$.

   By hypothesis, there is a chance of $\epsilon(n)$ that (m^*, t^*) is valid and m^* has never been requested to $\mathcal{O}_\Pi$, i.e., us. Now, we need to extract a valid (m', t') for Π'. There are three cases:

   - If the pair (r, l) has never been used in any of our requests, then $r||l||i||m_i^*$ has never been queried to $\mathcal{O}_{\Pi'}$, and we have our forgery: take any of $(r||l||i||m_i^*, t_i)$.
   - If (r, l) has been used before, then we need to distinguish between two cases. The first case is if r has been used two or more times before: as r is picked randomly, this event happens with probability $\frac{q(n)}{2^{n/4}}$. In that case, for correct l, $\mathcal{A}_\Pi$ could have just rearranged blocks from two previous messages m^1 and m^2 to build a new message m^*. For him, this is an existential forgery against Π. But for us, all those blocks $r||l||i||m_i^{1,2}$ have been queried to our oracle before, and so none of them are fresh, and this is not a forgery against Π'.
   - If (r, l) has been used before and r has only been used once, then there is a past message m^1 whose tag used this r. Let's compare the two messages:

$$m^1 \to m_1^1||m_2^1||\dots||m_d^1 \to \langle r, \mathsf{Mac}'_k(r||l||1||m_1^1), \dots \rangle$$
$$m^* \to m_1^*||m_2^*||\dots||m_d^* \to \langle r, \mathsf{Mac}'_k(r||l||1||m_1^*), \dots \rangle$$

     As m^* is a forgery against Π, $m^1 \neq m^*$, so $\exists j \mid m_j^1 \neq m_j^*$. For this i, the corresponding $(r||l||j||m_j^*, t_j)$ is a forgery against Π'.

If now we compute it,

$$\begin{aligned}\epsilon'(n) = \Pr[\mathsf{MacForge}_{\mathcal{A}_{\Pi'},\Pi'}(n) = 1] &\geq \Pr[\mathsf{MacForge}_{\mathcal{A}_{\Pi},\Pi}(n) = 1 \\ &\quad \wedge \text{ We can build a forgery from it}] \\ &\geq \Pr[\mathsf{MacForge}_{\mathcal{A}_{\Pi},\Pi}(n) = 1] \cdot \left(1 - \frac{q(n)}{2^{n/4}}\right) \\ &= \epsilon(n) \left(1 - \frac{q(n)}{2^{n/4}}\right)\end{aligned}$$

As we know that Π' is EUF-CMA, then $\epsilon'(n)$ must be negligible, so the whole right-hand side must be negligible too, and so $\epsilon(n)$ must be negligible.

Other said, if $\epsilon(n)$ is non-negligible, then the right-hand side is non-negligible (the term in parenthesis is negligibly close to 1), and so $\epsilon'(n)$ must be non-negligible too.

3.4: Exercise 3 (Authenticated encryption and sPRP)

Solution to Exercise 3.4

See Exercise 2 of APE4.

3.5: Exercise 4 (Authenticated encryption)

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an authenticated encryption scheme where $0 \notin \mathcal{C}$ (that is, the string "0" is not a possible ciphertext for Π). Consider the following scheme $\Pi' := (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ with:

- $\mathsf{Gen}' = \mathsf{Gen}$
- $\mathsf{Enc}' = \mathsf{Enc}$
- $\forall k : \mathsf{Dec}'(c) = \begin{cases} \mathsf{Dec}(c) \text{ if } c \neq 0, \\ 0 \text{ if } c = 0. \end{cases}$

1. Is Π' unforgeable?
2. Is Π' CCA secure?

Solution to Exercise 3.5

1. It is not unforgeable. To see this, simply build the following adversary $\mathcal{A}$:
   (1) The oracle-challenger chooses $k \leftarrow \mathcal{K}$.
   (2) We have no query to do to the oracle.
   (3) Output $c = 0$.

   By definition, $\mathsf{Dec}'_k(0) = 0 \, \forall k$, so it is a valid message. So $\Pr[\mathsf{EncForge}_{\mathcal{A},\Pi'}(n) = 1] = 1$.
2. It is CCA-secure (sure at 95%). To show this, let's do a reduction. If we have a PPT adversary $\mathcal{A}_{\Pi'}$ against Π', we can build an adversary $\mathcal{A}_{\Pi}$ against Π and run the following experiment (from the viewpoint of $\mathcal{A}_{\Pi}$):
   (1) The oracle-challenger chooses $k \leftarrow \mathcal{K}$.
   (2) First query phase:

- When $\mathcal{A}_{\Pi'}$ asks for $\mathsf{Enc}'_k(m)$ of a message m, we simply forward the request to $\mathcal{O}_\Pi$, receive $c = \mathsf{Enc}_k(m)$, and return it to $\mathcal{A}_{\Pi'}$
- When $\mathcal{A}_{\Pi'}$ asks for $\mathsf{Dec}'_k(c)$ of a ciphertext c, if $c = 0$, then we return 0, and if $c \neq 0$, we forward the request to $\mathcal{O}_\Pi$, receive $m = \mathsf{Dec}_k(c)$ and return it to $\mathcal{A}_{\Pi'}$.

(3) $\mathcal{A}_{\Pi'}$ outputs its messages m_0, m_1. We output them.

(4) The challenger-oracle picks $b \leftarrow \{0,1\}$, returns $c^* = \mathsf{Enc}_k(m_b)$, sends that to us, and we forward it to $\mathcal{A}_{\Pi'}$.

(5) Second query phase, similar to the first one, with the exception that $\mathcal{A}_{\Pi'}$ cannot ask for $\mathsf{Dec}'_k(c^*)$.

(6) $\mathcal{A}_{\Pi'}$ outputs its guess b', and we output $b'' = b'$.

The adversary $\mathcal{A}_{\Pi'}$ is exactly in the correct conditions, our adversary $\mathcal{A}_\Pi$ is PPT, and we output the same value, so $\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}_\Pi,\Pi}(n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}_{\Pi'},\Pi'}(n) = 1] = \Pr[b' = b] = \Pr[b'' = b] = \epsilon$ and, as we know that Π is CCA-secure, then we have that Π' is CCA-secure. This "zero-filling" has no impact on the CCA-security, only on the unforgeability.

3.6: Exercise 5

Let F be a PRF. Below, we describe three *insecure variable-length* message authentication codes (*a.k.a.* MACs), Π_1, Π_2 and Π_3, which all use the same key generation algorithm $\mathcal{G}$. The message space is *any (non negative) number* of message blocks in $\{0,1\}^n$, where n is the security parameter.

$\mathcal{G}(1^n)$ outputs a random key $k \leftarrow \{0,1\}^n$.

The scheme Π_3 is built from Π_2 which is itself built from Π_1 as an (unsuccessful) attempt to "patch" the previous scheme:

$\Pi_1 = (\mathsf{Gen}, \mathsf{Mac}^1, \mathsf{Vrfy}^1)$**:** *"Deterministic MAC – Chaining PRFs"*

$\mathsf{Mac}^1_k(m_1, \ldots, m_\ell)$ computes $t_1 = F_k(m_1)$ as well as $t_i = F_k(m_i \oplus t_{i-1})$, for $i = 2$ to ℓ, and returns $t \coloneqq t_\ell$ (note that only the last block is returned).

$\mathsf{Vrfy}^1_k((m_1, \ldots, m_\ell), t)$ outputs 1 if $\mathsf{Mac}^1_k(m_1, \ldots, m_\ell) = t$, and 0 otherwise.

$\Pi_2 = (\mathsf{Gen}, \mathsf{Mac}^2, \mathsf{Vrfy}^2)$**:** *"Padding a random message block in the end"*

$\mathsf{Mac}^2_k(m_1, \ldots, m_\ell)$ first picks a random $r \leftarrow \{0,1\}^n$ and then runs $t = \mathsf{Mac}^1_k(m_1, \ldots, m_\ell, r)$ and outputs (r, t).

$\mathsf{Vrfy}^2_k((m_1, \ldots, m_\ell), (r, t))$ outputs 1 if $\mathsf{Mac}^1_k(m_1, \ldots, m_\ell, r) = t$, and 0 otherwise.

$\Pi_3 = (\mathsf{Gen}, \mathsf{Mac}^3, \mathsf{Vrfy}^3)$**:** *"Padding a random message block in the beginning"*

$\mathsf{Mac}^3_k(m_1, \ldots, m_\ell)$ first picks a random $s \leftarrow \{0,1\}^n$ and then runs $(r, t) = \mathsf{Mac}^2_k(s, m_1, \ldots, m_\ell)$ and outputs (r, s, t).

$\mathsf{Vrfy}^3_k((m_1, \ldots, m_\ell), (r, s, t))$ outputs 1 if $\mathsf{Mac}^1_k(s, m_1, \ldots, m_\ell, r) = t$, and 0 otherwise.

1. Describe $\mathsf{Mac}^3_k(m_1, \ldots, m_\ell)$ explicitly in term of computations of F_k (and $\oplus$ of course).
2. Show the correctness of Π_3.
3. Mount a forgery attack on these MACs.

Solution to Exercise 3.6

1. For random $r, s \leftarrow \{0,1\}^n$, $\mathsf{Mac}^3_k(m_1, \ldots, m_\ell)$ computes $t_0 = F_k(s)$, then $t_i = F_k(m_i \oplus t_{i-1})$ for $i = 1$ to ℓ, and finally $t_{\ell+1} = F_k(r \oplus t_\ell)$. It outputs (r, s, t) where $t \coloneqq t_{\ell+1}$.
2. Using the description of Mac^1_k, on inputs $(s, m_1, \ldots, m_\ell, r)$ it gives $t'_1 = F_k(s)$, for $i = 1$ to ℓ,

$t'_{i+1} = F_k(m_i \oplus t'_i)$, and $t'_{\ell+2} = F_k(r \oplus t'_{\ell+1})$. $t'_{\ell+2}$ is the output, or equivalently:

$$\begin{aligned} F_k(r \oplus t'_{\ell+1}) &= F_k\Big(r \oplus F_k\Big(m_\ell \oplus F_k\big(m_{\ell-1} \oplus \dots F_k(m_1 \oplus F_k(s)) \dots\big)\Big)\Big) \\ &= t. \end{aligned}$$

3. These MACs are not even one-time secure:

   Π_1: (1) query $\mathsf{Mac}_k^1(m)$ on any $m \in \{0,1\}^n$ and get the tag $t = F_k(m)$;
   (2) output $((m, m \oplus t), t)$.
   $t_1 = F_k(m)$, $t_2 = F_k(m \oplus t \oplus t_1) = F_k(m \oplus 0) = t$.

   Π_2: (1) query $\mathsf{Mac}_k^2(m)$ on any $m \in \{0,1\}^n$ and get the tag (r, t), where $t = F_k(F_k(m) \oplus r)$;
   (2) output $((m, r, m \oplus t), (r, t))$.
   $t_1 = F_k(m)$, $t_2 = F_k(F_k(m) \oplus r) = t$, $t_3 = F_k(t \oplus m \oplus t) = F_k(m)$, $t_4 = F_k(F_k(m) \oplus r) = t$.

   Π_3: (1) query $\mathsf{Mac}_k^3(m)$ on any $m \in \{0,1\}^n$ and get the tag (r, s, t), where $t = F_k(F_k(F_k(s) \oplus m) \oplus r)$
   (2) output $((m, r, s \oplus t, m), (r, s, t))$.
   $t_1 = F_k(s)$, $t_2 = F_k(F_k(s) \oplus m)$, $t_3 = F_k(F_k(F_k(s) \oplus m) \oplus r) = t$, $t_4 = F_k(t \oplus s \oplus t) = F_k(s)$, $t_5 = F_k(F_k(s) \oplus m)$, $t_6 = F_k(F_k(F_k(s) \oplus m) \oplus r) = t$.

3.7: Exercise 6

Let F be a pseudorandom function, G be a pseudorandom permutation, T be a public n-bit constant, k be a n-bit secret key, m be a n-bit message, IV be a n-bit random value chosen by the party computing the encryption (resp. MAC) before each operation. Among the following constructions, determine the ones that would be acceptable and justify your answer. (Your justifications can rely on results that have been presented during the class.)

1. $\mathsf{Enc}_k(m) \coloneqq F_k(m \oplus T)$ as an encryption scheme secure against eavesdropping.
2. $\mathsf{Enc}_k(m) \coloneqq G_k(m \oplus T)$ as an encryption scheme secure against eavesdropping.
3. $\mathsf{Enc}_k(m) \coloneqq G_k(m \oplus T)$ as an encryption scheme secure against a CPA-adversary.
4. $\mathsf{Enc}_k(m) \coloneqq (IV, G_k(m \oplus T \oplus IV))$ as an encryption scheme secure against a CPA-adversary.
5. $\mathsf{Mac}_k(m) \coloneqq F_k(m \oplus T)$ as a MAC scheme existentially unforgeable under an adaptive chosen-message attack.
6. $\mathsf{Mac}_k(m) \coloneqq (IV, G_k(m \oplus IV \oplus T))$ as a MAC scheme existentially unforgeable under an adaptive chosen-message attack.

Solution to Exercise 3.7

1. No, decryption does not work in general, as we do not necessarily know how to invert a PRF.
2. Yes, it is secure. The reduction to the PRP security can work as follows: the reduction gets m_0 and m_1 from the eavesdropper adversary, picks a random bit b, sends $m_b \oplus T$ to the challenger and receives a value c that it sends back to the eavesdropper adversary. This adversary has a probability exactly $1/2$ to guess b if c comes from a random permutation, and a probability $1/2 + \epsilon$ to guess b if c comes from a pseudorandom permutation. The reduction can therefore claim that it sees a PRP every time the adversary makes a successful guess. The difference between the two probabilities is ϵ. So, if F is a PRP, ϵ must be negligible.
3. No: it is not probabilistic. CPA security is not achievable with deterministic encryption.

4. We first observe that the $\oplus T$ does not matter: since T is public, the adversary has the possibility to adapt its choice of m in order to cancel it. Now, if we abstract from T, this is exactly the CBC encryption mode, which we know to be CPA-secure.

5. Again, we observe that the $\oplus T$ does not matter: since T is public, the adversary has the possibility to choose exactly on which value F_k will be applied. Now, if we abstract from T, this is exactly the basic MAC scheme from the class, which is secure.

6. This is insecure. Given one tag (IV, t) on a message m, the adversary can produce a valid tag (IV', t) on the message $m' = m \oplus IV \oplus IV'$ for any IV'.

3.8: Exercise 7 (Blue-ray security)

The movie industry wants to protect digital content distributed on DVD's. We study one possible approach. Suppose there are at most a total of n DVD players in the world (e.g. $n = 2^{32}$). We view these n players as the leaves of a binary tree of height $\log_2 n$. Each node ν_j in this binary tree contains an AES key K_j such that $\text{Enc}_{K_j} : \{0,1\}^l \to \{0,1\}^l$ is assumed to be a *secure* encryption. These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player is assigned a serial number $i \in [0, n-1]$. Consider the set S_i of $\log_2(n+1)$ nodes along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the $\log_2(n+1)$ keys associated with the nodes in S_i. In this way each DVD player ships with $\log_2(n+1)$ keys embedded in it (these keys are supposedly inaccessible to consumers).

1. Since all DVD players have the key *root* (noted K_{root}), find a way to protect the content $M \in \{0,1\}^l$ of a DVD such that all players can decrypt the movie (and then read it).

2. Now suppose that a hacker has been able to extract the key K_{root} embedded in his DVD player and has published it on the Internet. Show how the movie industry can encrypt the contents of a new DVD $M \in \{0,1\}^l$ such that only the owners of a DVD player can read it. Note that the movie indutry does not want to produce several encryptions of the same content M *i.e.* there will be a single manner to protect the DVD.

3. Suppose the $\log_2 n$ keys embedded in DVD player number r are exposed by hackers and published on the Internet. Show that when the movie industry is about to distribute a new DVD movie they can encrypt the contents of the DVD using a ciphertext of size $l \cdot (1 + \log_2 n)$ so that all DVD players can decrypt the movie except for player number r. In effect, the movie industry disables player number r.

   Hint: the DVD will contain $\log_2 n$ *ciphertexts where each ciphertext is the encryption of a unique* K *under certain* $\log_2 n$ *keys from the binary tree.*

Solution to Exercise 3.8

1. Encrypt all the DVDs with K_{root}, i.e. build the ciphertext: $c_{root} = Enc_{K_{root}}(M)$

2. At the beginning of the DVD, encrypt a random key K twice (using K_2 and then K_3): $Enc_{K_2}(K)$ and $Enc_{K_3}(K)$.

   We then encrypt all the content M of the DVD using K. (We suppose K_1 is K_{root}) K_2 and K_3 are the keys associated to the 2 childs of the root.

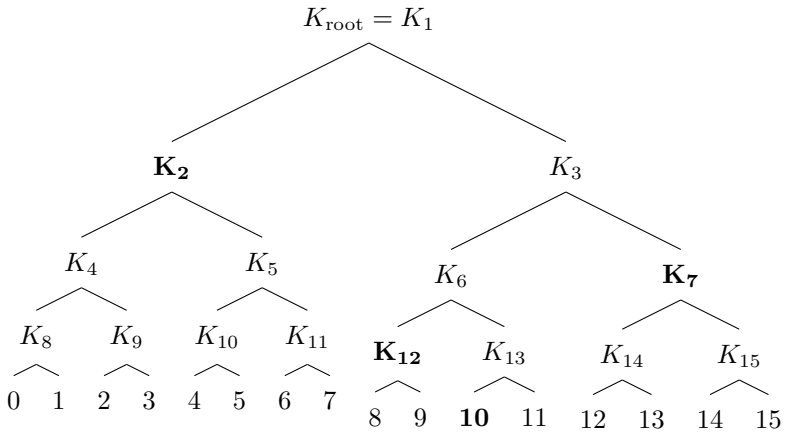
   Other possibility

   Simply encrypt the contents of the new DVD M as: $Enc_{K2}(M)||Enc_{K3}(M)$. This way, each DVD player is still able to decrypt the content, but nobody that only has K_{root}.

3. For every node i on the path from the root to r, we must add an encryption of K with the key of its child that is not in the path. For example, if $r = 10$ and $n = 16$, we must include the bold keys, so we will include the encryption of K_7 in a DVD which is quite something.

Expressed differently

If we want to disable the DVD player 11 for example, we can create a new ciphertext as: $c_{disable} = Enc_{K_2}(M)||Enc_{K_7}(M)||Enc_{K_{12}}(M)||Enc_{K_{player10}}(M)$



3.9: Exercise 8 (Authenticated encryption, or not)

Solution to Exercise 3.9

See Exercise 1 of APE10.

3.10: Exercise 9 (Authenticated encryption)

Solution to Exercise 3.10

See Exercise 1 of APE4.

3.11: Exercise X (Mode of operation)

Show formally that ECB-mode encryption does not have indistinguishable encryptions in the presence of an eavesdropper.

Solution to Exercise 3.11

Let say we have to split the message into m messages of n bits. Choosing $m_0 = M_0\|M_0\|\cdots\|M_0$ and $m_1 = M_1\|M_2\|\cdots\|M_m$ with $M_i \neq M_j$ for $i \neq j$, if $b = 0$, $\mathcal{A}$ will get $c = C_0\|C_0\|\cdots\|C_0$ for some $C_0 \in \mathcal{C}$. But if b=1, $\mathcal{A}$ will get $c = C_1\|C_2\|\cdots\|C_m$ for some $C_i \in \mathcal{C}$.

An adversary $\mathcal{A}$ can output $b = 0$ iff all the C_is are equals. We have

$$\Pr[\mathsf{PrivK}^{\text{eav}}_{\mathcal{A},\text{ECB}}(nm)] = \frac{1}{2} + \frac{1}{2} = 1$$

since the C_i cannot be equal if $b = 1$ since we use a PRP. If two different M_i were encrypted as same C_i, decryption wouldn't be possible.

APE 4

4.1: Exercise 1 (Authenticated encryption, August 2019 exam)

1. Let $F\colon \mathcal{K} \times \{0,1\}^n \mapsto \{0,1\}^n$ be a PRF. Consider the following Authenticated Encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follow:

   - Gen: pick a key uniformly at random in $\mathcal{K}$;
   - Enc: on input m and k:
     - Parse m in l blocks $m_1, \ldots, m_l$ with $|m_1| = \cdots = |m_l| = n$
     - Pick r uniformly at random in $\{0,1\}^{\frac{n}{2}}$
     - For $i = 1, \ldots, l$:
       $$c_i = F_k(r||i) \oplus m_i$$
     - $c_{l+1} = F_k(r||l+1) \oplus \left(\oplus_{i=1}^{l} m_i\right)$
     - Return (r, c) with $c = (c_1, \ldots, c_l, c_{l+1})$.
   - Dec consequently.

   [For simplicity we suppose that for every message all blocks are *full*, that is, when parsed the last block has length n (i.e., $|m_l| = n$).

   When we write r||i we mean that the number i is written in binary notation putting as many zeros on the left as necessary.]

   (a) Is Π unforgeable? Prove or confute[1].

   (b) Is Π CCA-secure? Prove or confute with an attack. [Hint: the previous answer may be useful...]

2. Let $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be an authenticated encryption scheme with binary messages of length n. For two binary vectors of length n we denote $\oplus$ the coordinate-wise XOR. 1_n denotes the all-1 vector of length n. Consider the following schemes:

   - $\Pi^1 \coloneqq (\mathsf{Gen}^1, \mathsf{Enc}^1, \mathsf{Dec}^1)$:
     - $\mathsf{Gen}^1 \coloneqq \mathsf{Gen}'$,
     - $\mathsf{Enc}^1_k(m) \coloneqq (c_1, c_2) = (\mathsf{Enc}'_k(m), \mathsf{Enc}'_k(m \oplus 1_n))$,
     - $\mathsf{Dec}^1_k((c_1, c_2)) \coloneqq \mathsf{Dec}'_k(c_1)$ if $\mathsf{Dec}'_k(c_1) \oplus \mathsf{Dec}'_k(c_2) = 1_n$, $\perp$ otherwise.
   - $\Pi^2 \coloneqq (\mathsf{Gen}^2, \mathsf{Enc}^2, \mathsf{Dec}^2)$:
     - $\mathsf{Gen}^2 \coloneqq \mathsf{Gen}'$,
     - $\mathsf{Enc}^2_k(m) \coloneqq \mathsf{Enc}'_k(m \oplus 1_n)$,
     - $\mathsf{Dec}^2_k(c) \coloneqq 1_n$ if $\mathsf{Dec}'_k(c) = \perp$, $\mathsf{Dec}'_k(c) \oplus 1_n$ otherwise.

   (a) Is Π^1 an authenticated encryption scheme? If not, explain which property you can break and how.

   (b) Is Π^2 an authenticated encryption scheme? If not, explain which property you can break and how.

Solution to Exercise 4.1

Note: in the following adversary descriptions, we skip the description of the unused phases of the security games.

1. Notice that there is a constraint on the size of $l+1$: $|l+1| \leq \frac{n}{2} \implies l < 2^{n/2} - 1$ and thus $|m| = n \cdot l < n \cdot (2^{n/2} - 1)$. This constraint doesn't play a role in the proofs however.

[1] *refute

(a) Obviously, for two messages of the same length m and m', we have that

$$\forall 1 \leq i \leq l\colon c'_i = m'_i \oplus F_k(r||i) = (m_i \oplus m'_i) \oplus m_i \oplus F_k(r||i) = (m_i \oplus m'_i) \oplus c_i.$$

And,

$$\begin{aligned} c'_{l+1} &= F_k(r||l+1) \oplus (\oplus_{i=1}^{l} m'_i) = F_k(r||l+1) \oplus (\oplus_{i=1}^{l} m_i) \oplus \left((\oplus_{i=1}^{l} m_i) \oplus (\oplus_{i=1}^{l} m'_i)\right) \\ &= c_{l+1} \oplus \oplus_{i=1}^{l}(m_i \oplus m'_i). \end{aligned}$$

So, we can build a forgery as follows:

i. Ask to the oracle for the encryption of a message, say, $m = 0_n$ (n times the bit 0) so that it consists of only one block, and receive the answer (r, c).

ii. Output (r, c^*) with $c^* = c \oplus (0_n \oplus 1_n) = c \oplus 1_n = c \oplus m^*$, which is the encryption of message $m^* = 1_n$.

By construction, it is a forgery with probability 1. So, the scheme is not unforgeable.

It is also possible to build a forgery by using a message such that $m_l = \oplus_{i=1}^{l-1} m_i$; then, $c_{l+1} = F_k(r||l+1) \oplus 0^n$, while $c_l = F_k(r||(l-1)+1) \oplus \oplus_{i=1}^{l-1} m_i$, and so we can send (r, c^*) with $c^* = (c_1, c_2, \ldots, c_l)$: we drop the last c_{l+1}.

(b) In a similar manner, we can build an adversary winning against the CCA game (adversary's viewpoint):

i. The challenger-oracle picks $k \coloneqq \mathsf{Gen}(1^n) \leftarrow \{0,1\}^n$ uniformly at random.

ii. Output the messages[a] $m^0 = 0_n$ and $m^1 = 1_n$, and get the challenge ciphertext $(r, c) = \mathsf{Enc}_k(m_b) = (r, c_1, c_2) = (r, F_k(r||1) \oplus m_b, F_k(r||2) \oplus m_b)$.

iii. Ask to the oracle the decryption of (r, c^*) where $c^* = (c_1 \oplus 0_{n-1}||1, c_2 \oplus 0_{n-1}||1)$. We get its answer as m^*. This is of course not the same ciphertext as c.

iv. If $m^* = 0_{n-1}||1$, output 0, else ($m^* = 1_{n-1}||0$), output 1.

By construction, in building c^* we have constructed the encryption of $m_b \oplus 0_{n-1}||1$, so we flipped the last bit of the encrypted message, allowing us to decrypt it. So the probability $\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] = 1$, the adversary is PPT, and we have broken CCA security.

Could we also break CPA security? No (90% sure), and we can proof it by reduction (distinguisher of g between PRF F_k and random function f, based on adversary A_Π against Π). Remark that, in the case that the "PRF" is a true random function, then each of the $g(r||i)$ are independent random values, and thus the c_i are also independent random value, and so no relation can be found between them in a single message (the ciphertext is just purely random), and the only attack is to hope for a reused r.

Note that the fact that c_{l+1} uses $F_k(r||l+1)$ and not $F_k(r||l)$ is important, because otherwise there would be a reuse of argument, and we can build an attack that even breaks eavesdropper security.

2. (a) Π^1 is not CCA-secure, and so is not an authenticated encryption scheme; our PPT adversary $\mathcal{A}$:

i. Key generation, as always.

ii. Output $m_0 = 0_n$ and $m_1 = 1_n$.
Receive the challenge $c = (c_1, c_2) = (\mathsf{Enc}'_k(m_b), \mathsf{Enc}'_k(m_b \oplus 1_n))$.

iii. Ask for the decryption of $c' = (c_2, c_1) \neq c$ and receive m'. Observe that

$$\begin{aligned} c' &= (\mathsf{Enc}'_k(m_b \oplus 1_n), \mathsf{Enc}'_k(m_b)) = (\mathsf{Enc}'_k(m_b \oplus 1_n), \mathsf{Enc}'_k((m_b \oplus 1_n) \oplus 1_n)) \\ &= \mathsf{Enc}^1_k(m_b \oplus 1_n) = \mathsf{Enc}^1_k(m_{1-b}). \end{aligned}$$

iv. Output 0 if $m' = m_1$, otherwise 1 ($m' = m_0$).

By construction, $\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi^1}(n) = 1] = 1$, which is a non-negligible advantage.

The scheme is also forgeable, in the same way: simply ask for encryption of m_0, receive c, then build c' as above: this is a valid ciphertext.

(b) Π^2 is forgeable, and so it not an authenticated encryption scheme; our PPT adversary $\mathcal{A}$:

i. Key generation, as usual.

ii. Output a random ciphertext c.

There are two cases:

- either c is the encryption of some message m by $\mathsf{Enc}'_k(\cdot)$, and so $\mathsf{Dec}^2_k(c) = m \oplus 1_n$, and thus it is a valid ciphertext;
- or c is not a valid ciphertext for Π', in which case $\mathsf{Dec}'_k(c) = \perp$, and thus $\mathsf{Dec}_k(c) = 1_n$, which means that c is also a valid ciphertext for Π^2.

Thus, in both cases, this random c is a valid ciphertext, and so $\Pr[\mathsf{EncForge}_{\mathcal{A},\Pi^2}(n) = 1] = 1$: the scheme is forgeable.

The fact that, when $\mathsf{Dec}'_k(c) = \perp$, we return 1_n instead of a more correct $\perp$, allows us to break the unforgeability.

Is the scheme CCA-secure? Other than this issue, Enc^2 is the same as Enc, and the only exploitable change in behaviour between Π' and Π^2 is the fact that, on Π' invalid encryptions, Π^2 returns 1_n. The only way an adversary could use this particularity would be if he asks for the decryption of an invalid ciphertext (generated at random by him, or by flipping a bit in a valid ciphertext), and he would always get the same 1_n answer, not very helpful. So the scheme is CCA-secure, and we can further prove it by doing a proof by reduction.

[a] We use the notation m^0 instead of m_0 to differenciate between a message and a message block.

4.2: Exercise 2 (Authenticated Encryption and sPRP)

Consider the following scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ based on the strong pseudorandom permutation $\mathsf{F}\colon \mathcal{K} \times \{0,1\}^n$, defined as follow:

- $\mathcal{M} = \{0,1\}^{\frac{n}{2}}$ (the message space)
- Gen picks a random key $k \in \mathcal{K}$
- $\mathsf{Enc}_k(m)$ picks a random value $r \in \{0,1\}^{\frac{n}{2}}$, and computes $c \coloneqq \mathsf{F}_k(m \| r)$
- $\mathsf{Dec}_k(c)$ computes $(m \| r) = \mathsf{F}_k^{-1}(c)$ and outputs m (the first half).

Answers the following questions:

- is Π unforgeable?
- is Π CCA-secure? (*To do at home*)
- is Π an authenticated encryption scheme? (*To do at home*)

Definition 1 (*Strong PseudoRandom Permutation*)
A function $\mathsf{F}\colon \mathcal{K} \times \mathcal{M} \mapsto \mathcal{M}$ is a (q,t,ϵ)- *strong pseudorandom permutation* (sprp) if for any (q,t)-bounded adversary, the advantage:

$$\mathsf{Adv}^{\mathsf{sprp}}_{\mathsf{adv}} \coloneqq \left| \Pr\left[\mathsf{adv}^{\mathsf{F}_k(\cdot),\mathsf{F}_k^{-1}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{adv}^{\mathsf{f}(\cdot,\cdot),\mathsf{f}^{-1}(\cdot,\cdot)} \Rightarrow 1\right] \right| \leq \epsilon$$

with k and f picked uniformly at random from their domains, respectively $\mathcal{K}$ and the set of permutations $\mathcal{M} \mapsto \mathcal{M}$.

Solution to Exercise 4.2

1. It is not unforgeable because since F_k is a PRP, it is bijective (i.e., every image has a pre-image). Then
$$\forall c \in \{0,1\}, \exists m, r \colon F_k^{-1}(c) = (m||r)$$
and $\Pr[\mathsf{EncForge}_{\mathcal{A},\Pi}(n) = 1] = 1$.

2. It is CCA-secure and we will prove it by reduction, assuming that the PRP F is strong. Thus, assume we have a PPT adversary $\mathcal{A}$ against the scheme Π with advantage $\epsilon_{\mathcal{A}}(n)$. Then, we can build a PPT distinguisher $\mathcal{D}$ between a sPRP and a random function, which plays the sPRP game as follows:

   (a) The challenger-oracle picks $k \coloneqq \mathsf{Gen}(1^n) \in \mathcal{K}$, and $b \leftarrow \{0,1\}$. If $b = 0$, then the challenger uses a true random function f for g, if $b = 1$, the challenger uses the sPRP $F_k(\cdot)$ for g.

   (b) First query phase:
   When $\mathcal{A}$ asks for the encryption of a message m, $\mathcal{D}$ picks $r \leftarrow \{0,1\}^{\frac{n}{2}}$ uniformly at random and queries its oracle on input $m||r$ obtaining, $c = g(m||r)$. $\mathcal{D}$ then forwards c to $\mathcal{A}$.
   When $\mathcal{A}$ asks for the decryption of ciphertext c, $\mathcal{D}$ queries its oracle on input c obtaining $m||r = g^{-1}(c)$, and $\mathcal{D}$ answers m to $\mathcal{A}$.

   (c) When $\mathcal{A}$ does the challenge query and outputs m_0^*, m_1^* with $|m_0^*| = |m_1^*| = \frac{n}{2}$ and $m_0^* \neq m_1^*$, $\mathcal{D}$ picks $r^* \leftarrow \{0,1\}^{\frac{n}{2}}$ uniformly at random, and $b' \leftarrow \{0,1\}$ uniformly at random too. $\mathcal{D}$ queries then its oracle on input $m_{b'}^*||r^*$ obtaining $c^* = g(m_{b'}^*||r^*)$. $\mathcal{D}$ forwards c^* to $\mathcal{A}$.

   (d) Second query phase, like the first one, except that $\mathcal{A}$ cannot ask for $\mathsf{Dec}_k(c^*)$.

   (e) At the end of the game, $\mathcal{A}$ outputs its guess b''. $\mathcal{D}$ outputs 1 iff $b'' = b'$, 0 otherwise: did the adversary $\mathcal{A}$ guess our correct pick of b'?.

   Let's compute the probability of success for $\mathcal{D}$:

   - If $b = 0$, then $g = f$, a random function, and so, each ciphertext generated by g and each decrypted message generated by g^{-1} is a random number, independent of each other. In this condition, the best thing the adversary $\mathcal{A}$ can do is wait for a collision on $m||r$ and thus $\Pr[\mathcal{D} \text{ outputs } 1 | b = 0] = \Pr[b'' = b'] = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \frac{q(n)}{2^{n/2}}$.
   - If $b = 1$, then $g = F_k$, and $\mathcal{A}$ is in the right conditions (its interface is respected) to have the advantage $\epsilon_{\mathcal{A}}$: $\Pr[\mathcal{D} \text{ outputs } 1 | b = 1] = \Pr[b'' = b'] = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \epsilon_{\mathcal{A}}$.

   Thus, the difference between the probabilities is
$$\epsilon_{\mathcal{D}}(n) = \left| \Pr[\mathcal{D}^{F_k(\cdot), F_k^{-1}(\cdot)}(1^n) = 1] - \Pr[\mathcal{D}^{f(\cdot), f^{-1}(\cdot)}(1^n) = 1] \right| = \left| \epsilon_{\mathcal{A}}(n) - \frac{q(n)}{2^{n/2}} \right|$$
   As we assume that F is a strong PRP, that the whole construction above is PPT, then $\epsilon_{\mathcal{D}}(n)$ must be negligible, and thus the right-hand side must be negligible, and thus $\epsilon_{\mathcal{A}}$ must be negligible. The scheme is thus CCA-secure.

3. Since the scheme Π is CCA-Secure but not unforgeable, it is not an authenticated encryption

4.3: Exercise 3 (Hash functions from... hash functions)

Let $H_2 \colon \{0,1\}^{2l} \mapsto \{0,1\}^l$ and $H_3 \colon \{0,1\}^{3l} \mapsto \{0,1\}^l$ be collision resistant hash functions. For $2l$-bit strings x_i's, consider the following two constructions.

- $H_4 \colon \{0,1\}^{4l} \mapsto \{0,1\}^l$; $x = x_1||x_2 \to H_2\left(H_2(x_1)||H_2(x_1 \oplus x_2)\right)$
- $H_6 \colon \{0,1\}^{6l} \mapsto \{0,1\}^l$; $x = x_1||x_2||x_3 \to H_3\left(H_2(x_1 \oplus x_2)||H_2(x_2 \oplus x_3)||H_2(x_3 \oplus x_1)\right)$

Determine whether these hash functions are still collision resistant or not.

Solution to Exercise 4.3

- Let's show that from a collision of H_4, we generate a collision for H_2 which prove that H_4 is collision resistant since H_2 is so. Let's suppose that we have $x_1\|x_2 \neq y_1\|y_2$ are such that $H_4(x_1\|x_2) = H_4(y_1\|y_2)$.
  - If $H_2(x_1)\|H_2(x_1 \oplus x_2) \neq H_2(y_1)\|H_2(y_1 \oplus y_2)$, we have a collision for H_2 since their image by H_2 is identical.
  - If $H_2(x_1)\|H_2(x_1 \oplus x_2) = H_2(y_1)\|H_2(y_1 \oplus y_2)$, we have $H_2(x_1) = H_2(y_1)$ *and* $H_2(x_1 \oplus x_2) = H_2(y_1 \oplus y_2)$.
    * If $x_1 \neq y_1$, we have a collision for x_2 since $H_2(x_1) = H_2(y_1)$.
    * If $x_1 = y_1$, then $x_2 \neq y_2$ since $x_1\|x_2 \neq y_1\|y_2$. Therefore $x_1 \oplus x_2 \neq y_1 \oplus y_2$ and we have collision on H_2.
- H_6 is not collision resistant since $H_6(x_1\|x_2\|x_3) = H_6((x_1 \oplus w)\|(x_2 \oplus w)\|(x_3 \oplus w))$ for all w (collision if $w \neq 0^{2l}$). Indeed, since $\oplus$ is associative and commutative,

$$
\begin{aligned}
&= H_6((x_1 \oplus w)\|(x_2 \oplus w)\|(x_3 \oplus w)) \\
&= H_3(H_2((x_1 \oplus w) \oplus (x_2 \oplus w))\|H_2((x_2 \oplus w) \oplus (x_3 \oplus w))\|H_2((x_3 \oplus w) \oplus (x_1 \oplus w))) \\
&= H_3(H_2(x_1 \oplus (w \oplus w) \oplus x_2)\|H_2(x_2 \oplus (w \oplus w) \oplus x_3)\|H_2(x_3 \oplus (w \oplus w) \oplus x_1))) \\
&= H_3(H_2(x_1 \oplus x_2)\|H_2(x_2 \oplus x_3)\|H_2(x_3 \oplus x_1))) \\
&= H_6(x_1\|x_2\|x_3).
\end{aligned}
$$

4.4: Exercise 4 (Block-cipher based hash function)

Considering a block cipher $E\colon \mathcal{K} \times \mathcal{M} \mapsto \mathcal{C};\ (k, m) \to E(k, m) = \mathsf{Enc}_k(m)$ with $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^l$, one may try to construct a collision resistant compression function from $\{0,1\}^{2l}$ to $\{0,1\}^l$. Show that the following methods do not work :

$$f_1(x, y) = E(y, x) \oplus y \quad \text{and} \quad f_2(x, y) = E(x, x) \oplus y$$

That is, show an efficient algorithm for constructing collisions for f_1 and f_2. Recall that the block cipher E and the corresponding decryption algorithm D are both known to you (and they are bijective functions).

Solution to Exercise 4.4

We will give 2 collisions for f_1 and f_2.

We know that $E(k, \cdot)$ is surjective because it must be injective and $\mathcal{M} = \mathcal{C}$. Therefore the decryption exists for all $c \in \mathcal{C}$! We will use it for the second collision of f_1.

For f_1, we have the 2 following collisions

$$
\begin{aligned}
f_1(D(E(y,x),y),E(y,x)) &= E(E(y,x),D(E(y,x),y)) \oplus E(y,x) \\
&= y \oplus E(y,x) \\
&= E(y,x) \oplus y \\
&= f_1(x,y) \\
f_1(D(0,E(y,x) \oplus y),0) &= E(0,D(0,E(y,x) \oplus y)) \oplus 0 \\
&= E(y,x) \oplus y \\
&= f_1(x,y).
\end{aligned}
$$

and for f_2 we have

$$\begin{aligned} f_2(x, E(x,x)) &= E(x,x) \oplus E(x,x) \\ &= 0 \qquad\qquad \forall x \in \{0,1\}^l \\ f_2(y, E(x,x)) &= E(y,y) \oplus E(x,x) \\ &= E(x,x) \oplus E(y,y) \\ &= f_2(x, E(y,y)). \end{aligned}$$

Other approach:

For f_1: Let's define k_1 and k_2 such that k_2 is equal to k_1 except for the last bit which is flipped. Let's now take an arbitrary x_1 for which we ask the encryption $T_{x_1} = E(k_1, x_1)$. Now let's flip the last bit of T_{x_1} and call the result T_{x_2}. We can now ask for the decryption of T_{x_2} given k_2 as input key, which we know exists since D is bijective and $\mathcal{C} = \{0,1\}^l$. We then obtain x_2. We can now observe that:

$$\begin{aligned} f_1(x_1, k_1) &= E(k_1, x_1) \oplus k_1 \\ &= T_{x_1} \oplus k_1 \\ f_1(x_2, k_2) &= E(k_2, x_2) \oplus k_2 \\ &= T_{x_2} \oplus k_2 \\ &= (T_{x_1} \oplus 0^{l-1}||1) \ \oplus \ (k_1 \oplus 0^{l-1}||1) \\ &= T_{x_1} \oplus k_1 \end{aligned}$$

For f_2: We can ask for $T_x = E(x,x)$ and $T_y = E(y,y)$ for two arbitrary (but different) x and y. We can see that:

$$\begin{aligned} f_2(y, T_x) &= E(y,y) \oplus T_x \\ &= T_y \oplus T_x \\ f_2(x, T_y) &= E(x,x) \oplus T_y \\ &= T_x \oplus T_y \end{aligned}$$

Which are both equal with different input, this is a collision.

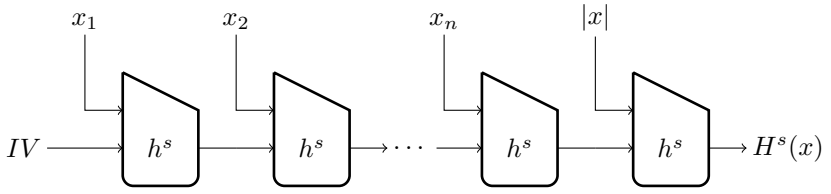
4.5: Exercise 5 (Authenticated encryption, or not)

Solution to Exercise 4.5

See Exercise 1 of APE10.

4.6: Exercise 6 (Variable-length MAC)

Considering a known hash function $h^s\colon \{0,1\}^{2l} \mapsto \{0,1\}^l$, let's note by H^s the corresponding Merkle-Damgård transform hash function, *i.e.*



when $x = x_1||\cdots||x_n$ for some integer n and when the x_i's are l-bit strings.

Show why, with a private key k of length l, the MAC scheme

$$t \coloneqq H^s(k||m),$$

is *not* existentially unforgeable under an adaptive chosen-message attack.

Solution to Exercise 4.6

If we have the tag of p, which is (let's consider that k and p are l bits long for simplicity)

$$t_p = H^s(k\|p) = h^s(h^s(h^s(IV\|k)\|p)\|2l)$$

we can find the tag of $p\|2l\|w$ (where w is l bits long for simplicity) without knowing k since we know h^s (it is public knowledge, only the secret k is secret and requires an oracle). It is

$$\begin{aligned} H^s(k\|p\|2l\|w) &= h^s(h^s(h^s(h^s(IV\|k)\|p)\|2l)\|w)\|4l) \\ &= h^s(h^s(H^s(k\|p)\|w)\|4l) \\ &= h^s(h^s(t_p\|w)\|4l) \end{aligned}$$

Since $p\|2l\|w \neq p$, this gives us an existential forgery.

4.7: Exercise 7 (Hash-MAC)

Suppose H_0 and H_1 are compression functions but only one is believed to be collision resistant. Besides, suppose MAC_0 and MAC_1 are message authentication codes but only one of the both schemes is known to be unforgeable. Is it possible to build a secure "hash-MAC" from these inputs? Justify your answer.

Solution to Exercise 4.7

We build $H(m) = H_0(m)\|H_1(m)$. If we have $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$ then $H_0(m_1) = H_0(m_2)$ and $H_1(m_1) = H_1(m_2)$ so the collision resistant hash function has a collision whichever it is. However, H is no more a compression function and we cannot use Merkle-Damgård.

The fact that we don't know the compressive factor also prevents us from building (simply) Merkle-Damgård or sponge constructions based on each of the functions.

The input of H therefore cannot have arbitrary length but its output is twice the length of the output of H_0 and H_1 so it is twice the size of a tag.

The output of Mac_0 and Mac_1 are the size of a tag so we can use the tag $H(\mathsf{Mac}_0(k,m)\|\,\mathsf{Mac}_1(k,m))$ for our Hash-MAC scheme. If we are able to output an existential forgery (m,t), since H is collision resistant, that means that we have found $\mathsf{Mac}_0(k,m)\|\,\mathsf{Mac}_1(k,m)$ and therefore we have found an existential forgery for both Mac_0 *and* Mac_1 which is absurd since one of them is believed to be unforgeable.

Our Hash-MAC scheme is therefore unforgeable.

4.8: Exercise 8 (Blue-ray security)

Solution to Exercise 4.8

See Exercise 7 of APE3.

APE 5

The TP5 of 2019-2020 reviewed the basics of number theory and group theory for this course. As they are not really useful for the exam and are fairly trivial (once the basics are know, that is), they are skipped in this document.

Below are the exercises of last year pertaining to groups and number theory.

5.1: Exercise 0 (Group order)

Solution to Exercise 5.1

See Exercise 5 of APE3.

5.2: Exercise 3 (Euclidean algorithm for gcd)

Let $a, b \in \mathbb{Z}$, $b \neq 0$, consider the following algorithm, presented in Algorithm 2. ($r = a\%b$ means that $a = qb + r$ where q is the quotient and r is the remainder).

Prove that x, the value returned by Algorithm 2, is $\mathsf{gcd}(a, b)$.

Hint:

- Prove that x divides $\mathsf{gcd}(a, b)$
- Prove that $\mathsf{gcd}(a, b)$ divides x

Input: a, b
Output: $\mathsf{gcd}(a, b)$
while $b \neq 0$ **do**
  $r \leftarrow a\%b$;
  $a \leftarrow b$;
  $b \leftarrow r$;
end
return (a)

Algorithm 2: The Euclidean gcd algorithm.

Solution to Exercise 5.2

According to the algorithm, we will have as successive value for the different remainder:

$$(r_2 = r_0\%r_1, r_3 = r_1\%r_2,\ r_4 = r_2\%r_3,\ \dots\ , r_n = r_{n-2}\%r_{n-1})$$

Where $r_0 = a$, $r_1 = b$ and r_n is the last non null remainder. Then we have the property that :

$$gcd(r_i, r_{i+1}) = gcd(r_{i+1}, r_{i+2})\ \forall i :\ 0 \leq i \leq n-2$$

Otherwise if it was not the case, $\exists i < n$ such that $r_i = 0$. But as r_n is the last non null remainder, we prove by contradiction this property.

As $gcd(r_{n-1}, r_n) = r_n$ because $r_n | r_{n-1}$ (since $r_{n+1} = r_{n-1}\%r_n = 0$), we can conclude that

$$gcd(r_0, r_1) = gcd(a, b) = gcd(r_{n-2}, r_{n-1}) = r_n$$

We have proved the value returned by the algorithm is the $gcd(a, b)$

5.3: Exercise 4

Consider the group $\mathbb{Z}_{17}^*$.

1. Compute 5^{-1}.
2. Compute 3^2, 3^3 and 3^4.
3. Does 3 generate the group?
4. Find $\log_7(11)$.

Solution to Exercise 5.3

Here because p is not too big, it is possible to evaluate "quickly" and "intuitively" the solutions. If it is too hard, there is an algorithm in the slides.

1. Because 35 mod 17 = 1, and $5 \cdot 7 = 35$.
   Then $5^{-1} = 7$. $(5 \cdot 7 = 1 \pmod{17})$
2. - $3^2 = 9 \pmod{17}$
   - $3^3 = 3^2 \cdot 3 = 27 \pmod{17} = 10 \pmod{17}$
   - $3^3 = 3^3 \cdot 3 = 30 \pmod{17} = 13 \pmod{17}$
3. According to *Fermat's little theorem*, if ord(g) = i then if $i|m = |G|$, where G is the commutative group. To see if 3 generate the group, we have to check if $3^i \neq 1$ where i are the divisor of $(p-1) = 16$ (except 16 of course !).
   - $3^1 = 3$ mod 17 (trivial)
   - $3^2 = 9$ mod 17 (evaluated previously)
   - $3^4 = 13$ mod 17 (evaluated previously)
   - $3^8 = (3^4)^2 = (13)^2 \pmod{17} = (-4)^2 \pmod{17} = 16$ mod 17

   We can see that 3 is a generator of the group.

   P.S. : The trick here is to remember the property of the modulo operation here in a Z_p^*:

   $$x = -(p - x) \pmod{p}$$

   It can make a lot of computing easier (it can become a real pain in the ass).
4. Here we have to find x such that:
   $$7^x = 11 \pmod{17}$$
   After (boring) computations, we have here :
   - $7^1 = 7$ mod 17
   - $7^2 = 15$ mod 17 = -2 mod 17
   - $7^3 = -14$ mod 17 = 3 mod 17
   - $7^4 = 4$ mod 17
   - $7^5 = 11$ mod 17 (Bingo)

   Then $log_7(11) = 5$

5.4: Exercise 5 (Group order)

In this exercise we consider the group $\mathbb{Z}_{59}^*$.

1. What is the order of 58?
2. What are the possible orders of an element of this group?
3. Find an element of order more than 20.

Solution to Exercise 5.4

The order of $g \in \mathbb{Z}_{59}^*$ is the smallest i where $g^i = 1$

1. ord(58) = 2 because:
   - 58^1 = 58 mod 59 = -1 mod 59
   - 58^2 = -58 mod 59 = 1 mod 59
2. According to *Fermat's little theorem*, the possible orders of a group $\mathbb{Z}_p^*$ are the divisor of p-1. Then, the possible orders are:
   - 1
   - 2
   - 29
   - 58
3. The best strategy here is to find a number g where

   $$ord(g) > 2$$

   (To assure you this is correct, just look at the possible ordrers).
   2 is a correct candidate.

APE 6

6.1: Exercise 1 (ElGamal Public Key Encryption and CCA Security)

1. Write the security definition of CCA security for a public key encryption scheme.
2. Let (c_1, c_2) and (c_1', c_2') be two ElGamal ciphertexts, of plaintext m and m' respectively. Can (c_1c_1', c_2c_2') be a ciphertext relatively to this scheme?
3. From (c_1, c_2) a ciphertext of m, can you build another ciphertext valid for m (remember that the public key is $(\mathbb{G}, g, q, h = g^x)$)? If yes, what is its decryption?
4. Show that ElGamal Public Key Encryption is not CCA secure.

Solution to Exercise 6.1

1. Given $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$, and adversary $\mathcal{A}$, define the following experiment $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}$:
   (a) Gen probabilistically selects $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$. pk / sk are the public/private key.
   (b) $\mathcal{A}$ is given oracle access to $\mathsf{Dec}_{sk}(\cdot)$.
   (c) $\mathcal{A}$ outputs m_0, m_1 of same length.
   (d) Choose $b \leftarrow \{0, 1\}$, and send $c \coloneqq \mathsf{Enc}_{pk}(m_b)$ to $\mathcal{A}$.
   (e) $\mathcal{A}$ is again given access to $\mathsf{Dec}_{sk}(\cdot)$ but he cannot ask $\mathsf{Dec}_{sk}(c)$.
   (f) $\mathcal{A}$ outputs b'.
   (g) Define $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) \coloneqq 1$ iff $b = b'$

   $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ is CCA-secure for a public key encryption scheme if $\forall$ PPT $\mathcal{A}, \exists \epsilon$:

$$\Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

2. (c_1c_1', c_2c_2') is a valid ciphertext for the message $m \cdot m'$. Indeed, $\mathsf{Enc}_{pk}(m \cdot m') = (g^{y''}, m \cdot m' \cdot h^{y''})$. If we set $y'' = y + y'$, then $= (g^y \cdot g^{y'}, m \cdot h^y \cdot m' \cdot h^{y'}) = (c_1c_1', c_2c_2')$. This is what we call an homomorphic encryption scheme: some operations on the ciphertexts have analogous operations on the encrypted plaintexts.
3. Yes. We have that $m = m \cdot 1$. To encrypt message 1, simply do $\mathsf{Enc}_{pk}(1) = (c_1', c_2') = (g^{y'}, 1 \cdot h^{y'})$ for some y' chosen randomly (do-able since this is a public encryption scheme), so that $c_1' \neq 1$ (simply don't take 0 as exponent). Then, by using the property proven before, we have that $(c_1^*, c_2^*) = (c_1c_1', c_2c_2')$ is a valid ciphertext for message m too.
4. Now, we are able to see why ElGamal does not hold CCA-security with the 2 properties found.

   If the adversary A, after received $c \coloneqq (c_1, c_2)$, compute $\mathsf{Enc}_{pk}(1) \coloneqq (c_1', c_2')$ and make the following oracle access $\mathsf{Dec}_{sk}(c_1c_1', c_2c_2') \coloneqq m_x$, he will now be able to (really) easily determine b from m_x. Then we can say that:

$$\Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\mathrm{ElGamal}}(n)] = 1$$

   Which means ElGamal does not have CCA-security with public key encryption.

   Other solution: We build an attacker A as follows :

(a) A outputs $m_0, m_1 \in M$.

(b) A receives $(c_1, c_2) = \mathsf{Enc}_{pk}(m_b) = (g^y, m_b \cdot g^{xy})$

(c) A queries the decryption of the transformed ciphertext $(c_1 \cdot g^{y'}, c_2 \cdot h^{y'})$ for some arbitrary chosen y' from $\mathbb{Z}_q$, and receives directly m_b as a response from his oracle.

6.2: Exercise 2 (A Variation of ElGamal in PKE)

Let consider ElGamal public encryption scheme with Encryption algorithm modified in the following way, where $\mathcal{M} = \{0, 1\}$:

- If $b = 0$ then choose a uniform $y \in \mathbb{Z}_q$ set $c_1 = g^y$, $c_2 = h^y$, and the ciphertext is (c_1, c_2).
- If $b = 1$ then choose independent uniform $y, z \in \mathbb{Z}_q$ set $c_1 = g^y$, $c_2 = g^z$, and the ciphertext is (c_1, c_2).

1. How is it possible to decrypt correctly such ciphertexts with the private key?
2. Show that this scheme is CPA secure if DDH holds in $\mathbb{G}$.

Solution to Exercise 6.2

1. We set:
$$\mathsf{Dec}_{sk}(c_1, c_2) = \begin{cases} 0 \text{ if } c_2 = c_1^x \\ 1 \text{ if } c_2 \neq c_1^x \end{cases}$$
If $b = 0$, then (c_1, c_2) is simply the ElGamal encryption of the message $m = 1$. In that case, $c_1^x = c_2$, ElGamal decryption always succeeds, and so $\Pr[\mathsf{Dec}_{sk}(c_1, c_2) = 0 | (c_1, c_2) \coloneqq \mathsf{Enc}_{pk}(0)] = 1$.
If $b = 1$, then (c_1, c_2) is the encryption of a message $\frac{c_2}{c_1^x}$. We have that
$$\Pr[\mathsf{Dec}_{sk}(c_1, c_2) = 0 | (c_1, c_2) = \mathsf{Enc}_{pk}(1)] = \Pr[c_2 = c_1^x | c_2 = g^z, c_1 = g^y] = \Pr[g^z = g^{xy}] = \frac{1}{|\mathbb{G}| = q}$$
and in contrast, $\Pr[\mathsf{Dec}_{sk}(c_1, c_2) = 1 | (c_1, c_2) \coloneqq \mathsf{Enc}_{pk}(1)] = 1 - \frac{1}{q}$. Overall, probability of correct decryption is
$$\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(b)) = b] = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot (1 - \frac{1}{q}) = 1 - \frac{1}{2q} \geq 1 - \frac{1}{2^{n+1}}.$$
So, there is a negligible probability of failure to decrypt correctly. This is allowed by the definition of a public-key encryption scheme, so decryption is OK.

2. We assume that DDH is difficult in $\mathbb{G}$. We can prove that the scheme is CPA-secure by reduction. We assume that we have a PPT adversary $\mathcal{A}$ against Π with advantage ϵ, and we build a distinguisher $\mathcal{D}$ as follows:

(1) Run $(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^n)$, such that $|q| = n$.
Choose uniformly at random $(x, y, z) \leftarrow \mathbb{Z}_q^3$.
Choose at random $b \leftarrow \{0, 1\}$.
Set $h_0 \coloneqq g^z$, $h_1 \coloneqq g^{xy}$.
Send $(\mathbb{G}, q, g, (g^x, g^y, h_b))$ to $\mathcal{D}$.

(2) $\mathcal{D}$ defines $pk = (\mathbb{G}, g, q, g^x)$ and sends it to $\mathcal{A}$

(3) $\mathcal{A}$ sends its challenge m_0, m_1 to $\mathcal{D}$. We assume w.l.o.g that $m_0 = 0$ and $m_1 = 1$.

(4) $\mathcal{D}$ sends $c = (g^y, h_b)$ to $\mathcal{A}$

(5) $\mathcal{A}$ outputs its guess b'. Then $\mathcal{D}$ outputs $b'' = 1 - b'$.

(6) Define $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n) = 1$ iff $b = b''$, and $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ iff b' is equal to the selected message.

We can observe that if $b = 0$, then message $m_1 = 1$ was sent to $\mathcal{A}$ as we sent $h_0 = g^z$, and if $b = 1$, then message $m_0 = 1$ was sent to $\mathcal{A}$, as we sent $h_1 = g^{xy}$.

If we run the experiment, we have:

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] &= \frac{1}{2} + \epsilon(n) \\ &= \Pr[b' \neq b] \\ &= \Pr[b'' = b] \\ &= \Pr[\mathsf{DDH}_{\mathcal{A}',\mathcal{G}}(n) = 1]\end{aligned}$$

If $\epsilon(n)$ is non negligible, $\mathcal{A}'$ is a PPT adversary that can break DDH with non negligible probability, which contradicts the assumption. Therefore $\mathcal{A}$ is CPA secure if DDH is hard in $\mathbb{G}$.

Note: there are a few problems in the computation of the probabilities; any fix / comment is welcome!

Another solution:

1. D outputs $1 \Leftrightarrow b' = 1$.
2. We observe:
   If $b = 0 \Rightarrow \Pr[D \text{ outputs } 1] = \frac{1}{2}$ (as g^z looks random in $\mathbb{G}$)
   If $b = 1 \Rightarrow \Pr[D \text{ outputs } 1] = \frac{1}{2} + \eta(n)$.
3. We thus have D distinguishing between g^z and g^{xy} with advantage $\eta(n)$. As we assume that DDH holds in $\mathbb{G}$, $\eta(n)$ must be negligible and this scheme is CPA-secure.

6.3: Exercise 3 (Decisional Diffie-Hellman, $\mathbb{Z}_p^*$, and QR_p)

Solution to Exercise 6.3

See Exercise 4 of APE7.

6.4: Exercise 4 (A variation of ElGamal in QR_p)

Let $p = 2q + 1$ with q prime, let $\mathbb{G} = QR_p$ the group of squares modulo p, and g be a generator of $\mathbb{G}$. We define ElGamal encryption scheme in this group: The private key is $(\mathbb{G}, g, q, x)$, the public key is $(\mathbb{G}, g, q, h = g^x)$ where $x \in \mathbb{Z}_q^*$ is chosen uniformly. To encrypt a message $m \in \mathbb{Z}_q$, choose a uniform $r \in \mathbb{Z}_q$, compute $c_1 = g^r \mod p$ and $c_2 = h^r + m \mod p$ and let the ciphertext be (c_1, c_2).

1. What is the order of g?
2. Is this scheme CPA-secure?

Solution to Exercise 6.4

1. As g is a generator of $\mathbb{G}$, then $\mathsf{ord}(g) = |\mathbb{G}|$. According to the previous exercise and given the fact that $\mathbb{G} := QR_p$, then $|\mathbb{G}| = \frac{p-1}{2} = q = \mathsf{ord}(g)$.
2. The scheme is CPA-Secure and we will prove it by assuming that QR_p holds in DDH.
   Suppose that we have a PPT adversary $\mathcal{A}$ which can solve ElGamal in QR_p with an advantage ϵ. It works like this:

(a) We send pk to $\mathcal{A}$ where pk $= (\mathbb{G}, g, q, h = g^x)$.

(b) $\mathcal{A}$ outputs m_0, m_1.

(c) The ElGamal challenger-oracle picks a random $b \leftarrow \{0,1\}$ and sends back $\mathsf{Enc}_{pk}(m_b) = (c_1, c_2)$ to $\mathcal{A}$.

(d) $\mathcal{A}$ then outputs b''. He wins if $b = b''$.

We will use $\mathcal{A}$ to build a distinguisher $\mathcal{D}$ able to solve $\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n)$:

(a) Run $\mathbb{G}(n)$ to obtain $(\mathbb{G}, q, g)$.

(b) We choose $(x, y, z) \leftarrow \mathbb{Z}_q^3$.

(c) We set $h_1 = g^{xy}$, $h_0 = g^z$ and $b \leftarrow \{0,1\}$. We forward then $((\mathbb{G}, q, g), (g^x, g^y, h_b))$ to $\mathcal{D}$.

(d) $\mathcal{D}$ will then build $pk = (\mathbb{G}, q, g, h_x \coloneqq g^x)$ and will forward pk to $\mathcal{A}$.

(e) $\mathcal{A}$ will then output its challenge queries m_0, m_1. $\mathcal{D}$ will pick $b' \leftarrow \{0,1\}$, and produce $c = (c_1, c_2) = (g^y, h_b + m_b \pmod p)$. We use $r \coloneqq y$ and use h_b as our value for $h_x^y = g^{xy}$. $\mathcal{D}$ sends c to $\mathcal{A}$.

(f) $\mathcal{A}$ will output b''.

(g) $\mathcal{D}$ will output $b''' \coloneqq 1 \Leftrightarrow b'' = b'$. We define $\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n) = 1$ iff $b''' = b$.

We analyse the chance of success to determine $\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n)$:

- if $b = 0$, then $\Pr[\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n) = 1] = \frac{1}{2}$: $\mathcal{A}$ is faced with random values, so he is not in the correct conditions.
- if $b = 1$, then $\Pr[\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n) = 1] = \frac{1}{2} + \epsilon(n)$: $\mathcal{A}$ operates correctly and its advantage is available.

Overall, we have

$$\Pr[\mathsf{DDH}_{\mathcal{D},\mathbb{G}}(n) = 1] = \frac{1}{2} + \frac{\epsilon(n)}{2}$$

As we assume that DDH is hard in QR_p, then it means that this probability should have a negligible advantage, and so $\epsilon(n)$ should be negligible: thus, the scheme that was attacked by $\mathcal{A}$ is CPA-secure.

6.5: Exercise 5 (DDH PRG)

Let $\mathbb{G}$ be a cyclic group of prime order q generated by $g \in \mathbb{G}$. Consider the following PRG defined over $(\mathbb{Z}^2 - q, \mathbb{G}^3)$: $G(\alpha, \beta) \coloneqq (g^\alpha, g^\beta, g^{\alpha\beta})$. Define what it means for a PRG over $(\mathbb{Z}^2 - q, \mathbb{G}^3)$ to be secure and show that G is a secure PRG assuming DDH holds in $\mathbb{G}$.

Solution to Exercise 6.5

Taking the definition, $G\colon \mathbb{Q}_q^2 \mapsto \mathbb{G}^3$ is a secure PRG over $\mathbb{G}$ if $\forall n$, $\forall$ PPT distinguisher $\mathcal{D}$, $\exists$ negl. ϵ such that:

$$|\Pr[D(x, y, z) = 1] - \Pr[D(G(\alpha, \beta)) = 1]| \leq \epsilon(n)$$

where $(x, y, z) \leftarrow \mathbb{G}^3$ and $(\alpha, \beta) \leftarrow \mathbb{Z}_q^2$, all randomly and uniformly and independent picked.

The other property of expansion is trivially verified by the definition itself (as $|\mathbb{Z}_q| = |\mathbb{G}| = q$).

We will proof the proposition by reduction.

Assume that we have a PPT adversary $\mathcal{D}$ that can distinguish between the PRG G as defined above and a random triplet of values with advantage ϵ. That is,

$$|\Pr[\mathcal{D}(x, y, z) = 1] - \Pr[\mathcal{D}(G(\alpha, \beta)) = 1]| \leq \epsilon(n)$$

Without loss of generality, we will assume that $\mathcal{D}$ outputs 1 when he identifies that he is in front of

the PRG. If that's not the case, simply reverse its output, and take $1-$ the probabilities above as the new probabilities of output, and the difference will be the same. Then, we can assume that

$$\Pr[\mathcal{D}(x,y,z)=1] = \Pr[\mathcal{D} \text{ outputs } 1|\ 3 \text{ random values}] = k$$

and

$$\Pr[\mathcal{D}(G(\alpha,\beta))=1] = \Pr[\mathcal{D} \text{ outputs } 1|\text{in front of PRG}] = k+\epsilon(n)$$

This is just a rewriting of the above condition. k is some value between 0 and 1, and we don't care what it is: we're just interested in the difference between the probabilities.

Then, let's build our PPT adversary $\mathcal{A}$ against DDH in $\mathbb{G}$. He will play the experiment $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)$:

1. Run $\mathcal{G}(1^n)$ to generate $(\mathbb{G},q,g)$ with g generator of $\mathbb{G}$ or order q and $|q|=n$.
   Choose $(x,y,z) \leftarrow \mathbb{Z}_q^3$, $b \leftarrow \{0,1\}$. Define $h_0 = g^z$, $h_1 = g^{xy}$.
   Send $(\mathbb{G},q,g,h_x \coloneqq g^x, h_y \coloneqq g^y, h_b)$ to $\mathcal{A}$.
2. Send to $\mathcal{D}(\mathbb{G},q,g)$ the following "challenge": $w \coloneqq (h_x,h_y,h_b)$.
   If $b=0$, this is (g^x,g^y,g^z), a triplet of pure random numbers.
   If $b=1$, this is (g^x,g^y,g^{xy}), the output of $G(x,y)$.
3. $\mathcal{D}$ outputs b'. Then, $\mathcal{A}$ outputs $b''=b'$. Define $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)=1$ iff $b''=b$.

Then, we just have to compute:

$$\begin{aligned}
\Pr[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)=1] &= \Pr[b''=b] \\
&= \Pr[b'=b] \\
&= \Pr[b'=1|b=1]\cdot\Pr[b=1] + \Pr[b'=0|b=0]\cdot\Pr[b=0] \\
&= (k+\epsilon(n))\cdot\frac{1}{2} + (1-k)\cdot\frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2}\epsilon(n) \leq \frac{1}{2}+\epsilon'(n).
\end{aligned}$$

As we assume that DDH is hard in $\mathbb{G}$, we assume that ϵ' is negligible, and so ϵ must be negligible too. This implies that $\mathcal{D}$ cannot distinguish between the PRG and random numbers efficiently, and thus that our PRG is secure.

6.6: Exercise X (A variant of ElGamal Encryption.)

Let us consider the following variant of ElGamal encryption. Let

- Gen output a pair $\langle pk, sk\rangle \coloneqq \langle(\mathsf{Gr},q,g,h),(\mathsf{Gr},q,g,x)\rangle$ as in traditional ElGamal encryption, except that x is selected in $\mathbb{Z}_q - \{0\}$;
- $\mathsf{Enc}_{pk}(m) \coloneqq \langle m\cdot g^y, h^y\rangle$ with $y \leftarrow \mathbb{Z}_q$ and $m \in \mathsf{Gr}$.

1. Define the corresponding decryption operation.
2. Why did we exclude "0" from the set in which x is selected?
3. Prove that this variant of ElGamal is CPA-secure if the DDH problem is hard with respect to the group key generation algorithm Gr.

Solution to Exercise 6.6

1. $\mathcal{D}_{sk}(c_1,c_2) = \frac{c_1}{c_2^{1/x}}$, where $1/x$ is the inverse of x mod q.

2. We exclude "0" as $0 \notin \mathbb{Z}_q^*$, so it has no inverse mod q.

3. The reduction A' proceeds as follows, using an attacker A of the modified ElGamal scheme:

   (a) A' starts an instance of A and gets $\mathbb{G}, q, g, g^x, g^y, g^z$ from the DDH challenger (where z is or is not xy)

   (b) A' checks if $g^x = 1$ (that is, if $x = 0$). If it is the case, it claims that it received a DDH tuple if $g^y = g^z$ (by returning 1) and a random tuple otherwise (by returning 0) and stops. The claim will be always correct, but this event will only happen with probability $1/q$.

   (c) A' forwards $(\mathbb{G}, q, g, g^x)$ as ElGamal public key to A'.

   (d) When A' outputs two messages m_0, m_1, A' flips a coin b and returns $(m_b g^y, g^z)$.

   (e) When A outputs its guess b', A' claims that it received a DDH tuple if and only if $b = b'$, and returns "1" in this case.

   If A runs in PPT, then so does A' (its extra operations are clearly PPT). Let X be the event that A' outputs 1 when receiving a DDH tuple, and Y be the event that A' outputs 1 when receiving a random tuple. Let us assume that A wins the CPA game with probability $\frac{1}{2} + \epsilon(n)$.

   We can see that $\Pr[X] = \Pr[X \wedge (x = 0)] + \Pr[X \wedge (x \neq 0)] = \frac{1}{q} + \frac{q-1}{q}(\frac{1}{2} + \epsilon(n))$. Indeed, when $x \neq 0$, A exactly sees inputs that are distributed exactly as he expects them. Furthermore, $\Pr[Y] = \Pr[Y \wedge (x = 0)] + \Pr[Y \wedge (x \neq 0)] = 0 + \frac{q-1}{q}\frac{1}{2}$. Indeed, when $x \neq 0$, the inputs of A are independent of b, and therefore A' wins with probability exactly $\frac{1}{2}$.

   As a result, $|\Pr[X] - \Pr[Y]| = \frac{1}{q} + \frac{q-1}{q}\epsilon(n)$. If $\epsilon(n)$ is non negligible (*i.e.* this ElGamal variant is non CPA secure), then $|\Pr[X] - \Pr[Y]|$ is non negligible and DDH does not hold in Gr. Reciprocally, if DDH holds in Gr, this ElGamal variant is CPA secure.

APE 7

7.1: Exercise 1 (Commitment scheme)

Define the bit-commitment scheme $\langle \mathcal{G}, \mathsf{Com}, \mathsf{Open} \rangle$ with the following PPT algorithms:

- $\mathsf{Gen}(1^n)$ sets pk as (PRG, R), where
  - G is a random generator $\{0,1\}^n \longmapsto \{0,1\}^{3n}$
  - R is a random $3n$-bit string
- $\mathsf{Com}_{pk}(b)$ with $b \in \{0,1\}$ provides (c, d) where:
  - Y is an n-bit string
  - if $b = 0$ $c = \mathsf{G}(Y)$
  - if $b = 1$, $c = \mathsf{G}(Y) \oplus R$
  - $d = (b, Y)$
- $\mathsf{Open}_{pk}(c, d)$ outputs b if it can recompute c from d and pk, or $\perp$ otherwise

1. Is this scheme perfectly hiding?
2. Is this scheme computationaly binding?
3. If the committer choose R is the scheme secure?

Solution to Exercise 7.1

1. For a scheme to be perfectly hiding, we need that $\forall \mathcal{A}$:

$$\Pr[\mathsf{Com}^{\mathsf{hide}}_{\mathcal{A},\Pi} = 1] = \frac{1}{2} \Leftrightarrow \Pr[c|b = 0] = \Pr[c|b = 1]$$

   If $b = 0$, then c has as much randomness as G, which has as much randomness as Y, so n bits of randomnes (=there are 2^n possible values for c).

   If $b = 1$, then c has as much randomness as G and R, so basically $3n$ bits of randomness (=there are 2^{3n} possible values for c).

   If we have unbounded computational power, then we could enumerate all possible outputs $\mathsf{G}(Y)$, and see if c is in this set of values. If $b = 0$, we are sure they are in; if $c = 1$, there are 2^{2n} possible R such that $\mathsf{G}(y) \oplus R$ cannot be distinguished from $\mathsf{G}(Y')$ (simply, $R = \mathsf{G}(Y) \oplus \mathsf{G}(Y')$), so there is a probability $\frac{2^{2n}}{2^{3n}} = \frac{1}{2^n}$ that $c \in \{\mathsf{G}(Y)\}$. So the probability of success for an unbounded adversary is

$$\frac{1}{2} \cdot 1 + \frac{1}{2}(1 - \frac{1}{2^n}) = 1 - \frac{1}{2^{n+1}}$$

   So, an unbounded adversary has near-certainty of breaking the hiding property.

   To break the hiding property, an adversary would need to enumerate all possible outputs of G (requires 2^n steps) if $G(Y) \oplus R \notin \{x | \forall u : x = G(u)\}$.

   But this kind of event has a negligible chance of probability ($\Pr = \frac{|\mathsf{G}(Y)|}{|\mathsf{G}(Y) \oplus R|} = \frac{2^n}{2^{3n}} = \frac{1}{2^{2n}} = \epsilon(n)$) so an adversary with an unbounded power of calculation can easily break the property of perfectly hiding.

   It is however simple to prove that the scheme is computationally hiding.

2. For the scheme to be computationally binding, it should be intractable to find (c, d_0, d_1) such that $\mathsf{Open}_{pk}(c, d_0) = 0$ and $\mathsf{Open}_{pk}(c, d_1) = 1$. If we replace, we find that it is equivalent to find

Y_0, Y_1 such that

$$c = \mathsf{G}(Y_0) = \mathsf{G}(Y_1) \oplus R$$

For this to be possible at all, we need to have $R = \mathsf{G}(Y_0) \oplus \mathsf{G}(Y_1)$ for some Y_0, Y_1. But, we have that $|\{R\}| = 2^{3n}$ while $|\{\mathsf{G}(Y_0) \oplus \mathsf{G}(Y_1)\}| \leq 2^{2n}$, so the probability of R being correct for this to happen is at most $\frac{2^{2n}}{2^{3n}} = \frac{1}{2^n}$, which is negligible. And so,

$$\Pr[\mathsf{Com}^{\mathsf{bind}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2^n} \quad \forall \mathcal{A}.$$

So even if we have an adversary capable of finding Y_0 and Y_1 with near-certainty, the fact that R can just be badly chosen for him causes its probability of success to be *in all cases* negligible. So, the scheme is computationally binding.

3. Not secure, because if the committer chooses $R = G(Y)$, then the opposite player can easily deduce the value of b. If $c = 0$, $b = 1$, else $b = 0$.

7.2: Exercise 2 (Commitment with DL)

Let $(\mathbb{G}, \cdot)$ be a group in which the discrete logarithm is difficult, with $|\mathbb{G}| = q$. Let g be a generator of the group and h be a random element of the group ((g, h) may be seen as the key of the hash function). Define the following hash funtion $\mathsf{H}\colon \mathbb{Z}_q^* \times \mathbb{Z}_q^* \mapsto \mathsf{G}$:

$$\mathsf{H}_{g,h}(\alpha, \beta) \coloneqq g^\alpha h^\beta$$

Prove that if the DL is difficult, then, the hash function is collision resistant. For simplicity we assume that q is prime.

Solution to Exercise 7.2

I don't know, maybe a reduction might be useful? It's been such a long time!

For the reduction, let's assume that we have an adversary $\mathcal{A}$ that can break the collision resistance by finding a collision with advantage $\epsilon(2n)$. We have $2n$ instead of n because the seed of the hash function has $2n$ bits instead of n. Then, we can build an adversary $\mathcal{D}$ that can solve the DL problem:

1. Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$ where g generates $\mathbb{G}$ of order q with $|q| = n$.
2. Choose $h \leftarrow \mathbb{G}$.
3. Send $(\mathbb{G}, q, g, h)$ to $\mathcal{D}$.
4. $\mathcal{D}$ uses $\mathcal{A}$: he sends (g, h) as the seed. Then, with probability $\epsilon(2n)$, $\mathcal{A}$ answers with (α, β) and (α', β') such that $\alpha \neq \alpha' \vee \beta \neq \beta'$ and

$$g^\alpha h^\beta = g^{\alpha'} g^{\beta'}$$

From this, if we want to find x such that $g^x = h$, then we just replace:

$$\begin{aligned} g^\alpha (g^x)^\beta &= g^{\alpha'} (g^x)^{\beta'} \\ \alpha + x\beta &= \alpha' + x\beta' \\ x &= \frac{\alpha' - \alpha}{\beta - \beta'} \end{aligned}$$

We have

$$\Pr[\mathsf{DLog}_{\mathcal{D},\mathcal{G}}(n) = 1] = \Pr[g^x = h] = \Pr[\mathsf{HashColl}_{\mathcal{A},\mathsf{H}}(n) = 1] = \epsilon(2n) \leq \epsilon'(n)$$

And, as we know that the DL problem is hard is $\mathbb{G}$, then we know that these probabilities should be negligible, and so that finding a collision is also hard.

7.3: Exercise 3 (Commitment scheme and batching)

Solution to Exercise 7.3

See Exercise 0 of APE9.

7.4: Exercise 4 (Decisional Diffie-Hellman and $\mathbb{Z}_p^*$)

The goals of this exercise are to define QR_p, prove some of its properties, and to show that in some groups DDH and CDH assumptions are conjectured not equivalent, as DDH is easy whereas CDH is conjectured to be hard.

1. For all element a of $\mathbb{Z}_{11}^*$, compute $a^2 \mod 11$.

   For a prime number p, we denote QR_p the set $\{x \in \mathbb{Z}_p^* \mid \exists a \in \mathbb{Z}_p^*, a^2 = x\}$, such x are called quadratic residues modulo p. Show that if p is odd then $|QR_p| = \frac{p-1}{2}$.

2. Show that, if p is odd, QR_p is a cyclic group (therefore, QR_p is a subgroup of $\mathbb{Z}_p^*$).

3. For all element a of $\mathbb{Z}_{11}^*$, compute $a^5 \mod 11$. Show that for any odd prime p, $x \in QR_p \Leftrightarrow x^{\frac{p-1}{2}} = 1 \mod p$, and that $x \notin QR_p \Leftrightarrow x^{\frac{p-1}{2}} = -1 \mod p$.

4. Show that 2 is a generator of $\mathbb{Z}_{11}^*$. For the following pairs (a, b), compute g^a, g^b and g^{ab} in $\mathbb{Z}_{11}^*$ where $g = 2$:

   - $(2, 8)$,
   - $(1, 4)$,
   - $(3, 5)$.

   Show that for p an odd prime, $g^{ab} \notin QR_p \Leftrightarrow g^a \notin QR_p$ and $g^b \notin QR_p$.

5. Show that DDH does not hold in $\mathbb{Z}_p^*$ with p an odd prime.

Solution to Exercise 7.4

An official solution was given in the exercise session.
For this exercise we will work with $\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

1. For all element a of $\mathbb{Z}_{11}^*$, I've calculated $a^2 \mod 11$.

   $$1^2 = 1 \quad 2^2 = 4 \quad 3^2 = 9 \quad 4^2 = 5 \quad 5^2 = 3 \quad 6^2 = 3 \quad 7^2 = 5 \quad 8^2 = 9 \quad 9^2 = 4 \quad 10^2 = 1$$

   We see that with p odd, we have $|QR_p| = \frac{p-1}{2}$. We can show it with this development:

2. Sadly, this sub-question does not yet have a solution. You are encouraged to submit one to the following address

   `https://github.com/gp2mv3/Syntheses`

   or by mail.

3. For all element a of $\mathbb{Z}_{11}^*$, I've calculated $a^5 \mod 11$.

$$1^5 = 3^5 = 4^5 = 5^5 = 9^5 = 1 \qquad 2^5 = 6^5 = 7^5 = 8^5 = 10^5 = 10$$

We can see that for p prime, we have $x \in QR_p \Leftrightarrow x^{\frac{p-1}{2}} = 1 \mod p$ and $x \notin QR_p \Leftrightarrow x^{\frac{p-1}{2}} = p-1 \mod p$.

- $x \in QR_p \Leftrightarrow x^{\frac{p-1}{2}} = 1 \mod p$:
  We know that
  $$x \in QR_p \Leftrightarrow \exists a \text{ s.t. } x = a^2 \mod p$$
  So we have now
  $$x = a^2 \mod p \Leftrightarrow x^{\frac{p-1}{2}} = 1 \mod p$$
  If we replace x by a we obtain $a^{2(\frac{p-1}{2})} = 1 \mod p$.
  But also more simply $a^{p-1} = 1 \mod p$ which is true by the group theory.
- $x \notin QR_p \Leftrightarrow x^{\frac{p-1}{2}} = p-1 \mod p$:
  We know that
  $$x \notin QR_p \Leftrightarrow \exists a \text{ s.t. } x = a^{1+2n} \mod p$$
  So we have now
  $$x = a^{1+2n} \mod p \Leftrightarrow x^{\frac{p-1}{2}} = -1 \mod p$$
  We replace x by a and we get
  $$a^{\frac{p-1}{2}} a^{n(p-1)} \mod p = -1 \mod p$$
  We know that $a^{n(p-1)} \mod p = 1$, thus we simplify the equation like
  $$a^{\frac{p-1}{2}} \mod p = -1 \mod p$$
  We know that $g = a^{\frac{p-1}{2}} \mod p \neq 1 \mod p$ but $g^2 = a^{p-1} = 1 \mod p$. The only solution of these two equations is $g = -1 \mod p$ which is equivalent to
  $$x^{\frac{p-1}{2}} = p-1 \mod p$$

4. The number 2 is a generator of $\mathbb{Z}_{11}^*$, because ord(2) = 10. In fact, we have $2^1 = 2$, $2^2 = 4$, $2^5 = 10$ and $2^10 = 1$. (Fermat's little theorem) We have $g = 2$ so:
   - $(2, 8)$: $g^2 = 4$, $g^8 = 3$ and $g^{16} = -2$
   - $(1, 4)$: $g^1 = 2$, $g^4 = 5$ and $g^4 = 5$
   - $(3, 5)$: $g^3 = -3$, $g^5 = -1$ and $g^{15} = -1$ TODO

   We have to show that $g^{ab} \notin QR_p \Leftrightarrow g^a \notin QR_p$ and $g^b \notin QR_p$.
   We know by the definition of the QR_p set that
   $$g^n \notin QR_p \Leftrightarrow \exists m \text{ s.t. } n = 2m + 1$$
   We can thus extract from $g^{ab} \notin QR_p$ that $\exists m$ s.t. $ab = 2m + 1$.
   So we are assured that a and b are not pairs, so we have the relation $g^a \notin QR_p$ and $g^b \notin QR_p$ if and only if $ab = 2m + 1$ which is equivalent to $g^{ab} \notin QR_p$. That was what we had to proof.

5. We have to show that DDH does not hold in $\mathbb{Z}_p^*$ with p an odd prime number.

   We define an attacker that can see p, g, g^a, g^b and receive $h_b = g^{ab}$ or g^z. The behaviour of the attacker will be this one:

- It receives $g^a \notin QR_p$ and $g^b \notin QR_p$:
  It will answer in function of h_b:
  - $h_b \notin QR_p$:
    It answers $h_b = g^{ab}$
  - $h_b \in QR_p$:
    It answers $h_b = g^z$
- It receives $g^a \in QR_p$ or $g^b \in QR_p$:
  It answers randomly.

We can identify four cases with their chances of success and appearance (we already know that $|QR_p|$ is of size $\frac{p-1}{2}$):

a) $g^a \in QR_p$ or $g^b \in QR_p$ appears 3/4 of the time with success = 1/2.

b) $g^a \notin QR_p$ and $g^b \notin QR_p$ with $h_b = g^{ab}$ appears 1/8 of the time with success = 1.

c) $g^a \notin QR_p$ and $g^b \notin QR_p$ with $h_b = g^z$ and $g^z \in QR_p$ appears 1/16 of the time with success = 1.

d) $g^a \notin QR_p$ and $g^b \notin QR_p$ with $h_b = g^z$ and $g^z \notin QR_p$ appears 1/16 of the time with success = 0.

We can now recalculate the expected value of success of our attacker:

$$\begin{aligned}\mathbb{E}(success) &= \frac{3}{4} \cdot \frac{1}{2} + \frac{1}{8} \cdot 1 + \frac{1}{16} \cdot 1 + \frac{1}{16} \cdot 0 \\ &= \frac{1}{2} + \frac{1}{16}\end{aligned}$$

This attacker has one sixteenth of probability more than one half which is not a negligible function (in fact it is a constant function). It is not DDH secure.

7.5: Exercise 5

Solution to Exercise 7.5

See Exercise 1 of APE8.

APE 8

8.1: Exercise 1 (Zero knowledge Petersen)

We work in a group $\mathbb{G}$ of prime order q with generator g. The Schnorr protocol, used to prove the knowledge of discrete logarithm, is (honest-verifier) zero-knowledge. However, the value $y = g^x \pmod p$ (for a safe prime $p = 2q + 1$) leaks some information about the discrete logarithm x (since for a given generator g of order q there is exactly one such x in $\mathbb{Z}_q$). On the other hand, the Pedersen commitment is perfectly hiding and thus does not reveal information about the committed value. The following protocol attempts to merge the both properties i.e., to prove the knowledge of a commited value under the Pedersen commitment scheme in a zero-knowledge manner.

The protocol. The public inputs of the proof are the prime p, the Pedersen public key (g, h), a security parameter k and a (hypothetic) commitment $c \in QR(p)$. The prover's private intputs are x and r in $\mathbb{Z}_q$ s.t. $c = g^x h^r \pmod p$. The protocol executes as follows.

- The prover randomly chooses $y, s \in_R \mathbb{Z}_q$ and sends $d = g^y h^s \pmod p$ to the verifier.
- The verifier randomly chooses $e \in_R \{0,1\}^k$ and sends it to the prover.
- The prover computes $z = y - ex$ and $t = s - er$ modulo q and sends it to the verifier.
- The verifier accepts the proof iff $d = c^e g^z h^t \pmod p$.

If the verifier accepts the proof, we say that the conversation $\langle d, e, (z, t)\rangle$ is valid.

1. Prove the correctness property of this construction.
2. Assume that an adversary is able to produce two valid responses for two distinct challenges, under the same commit. How can you use this faculty to extract an opening of c? Discuss the soundness property of the protocol.
3. Assume you are able to "rewind" an adversarial prover who tries to build a valid conversation. How can you use this faculty to extract an opening of c. Which property did you break ? Briefly discuss the soundness property of the protocol.
4. Show how a valid conversation $\langle d, e, (z, t)\rangle$ can be simulated from c, without the use of any private inputs. (Assume that the valid conversation involves honest parties.)
5. Generalize the process to prove the knowledge of an opening to a multi-Pedersen commitment as in exercise 3.

Solution to Exercise 8.1

1. The construction is correct if $Pr[d \neq c^e g^z h^t] \leq \epsilon(n)$. Let's evaluate this probability:

$$\Pr[d \neq c^e g^z h^t] = \Pr[g^y h^s \neq (g^x h^r)^e g^{y-ex} h^{s-er}] = Pr[g^y h^s \neq g^y h^r] = 0$$

   Then our construction is correct.

2. When using the same commitment $c = g^x h^r$, if we get two different conversations (e, z, t) and $e', z', t')$, we can recover the private secret (x, r) by doing the following:

$$\begin{cases} z = y - ex \\ z' = y - e'x \end{cases} \Rightarrow x = \frac{z - z'}{e' - e}$$

$$\begin{cases} t = s - er \\ t' = s - e'r \end{cases} \Rightarrow r = \frac{t - t'}{e' - e}$$

This means that the adversary P^* should know x and r, otherwise he could not build two valid responses for the same commit.

Does that break the zero-knowledge property? In a sense, no, as in a practical scheme, the commit would be different each time due to a random r.

3. (This answer has not been verified, and looks wrong, but as the subquestion has not been asked this year, I can't verify it.)

   When we get the conversation $(d, e, (z, t))$, we can "rewind" the conversation to submit another e' and get new z' and t'. Therefore we can obtain the private key (x,r) by doing those calculations:

$$\begin{cases} z = y - ex \\ z' = y - e'x \end{cases} \Rightarrow x = \frac{z - z'}{e' - e}$$

$$\begin{cases} t = s - er \\ t' = s - e'r \end{cases} \Rightarrow r = \frac{t - t'}{e' - e}$$

   We broke the zero-knowledge property since the verifier can extract the private key using such power.

   According to the assistants, since it is not zero-knowledge, there is no point of discussing the soundness property.

4. It is easy to show, with honest parties, how we can simulate from c a new valid conversation:

   (a) We pick e, z and t randomly.

   (b) We evaluate $d \coloneqq c^e z^z h^t$.

5. To generalize the process, we have:

   - pk = g_1^x, ..., g_n^x, h
   - sk = x_1, ..., x_n, r
   - c = $g_1^x \cdot ... \cdot g_n^x \cdot h^r$

8.2: Exercise 2 (Schnorr ZKP with faulty PRG)

Let us study what happens when the prover of a Schnorr ZKP uses a faulty random generator. this generator is used to choose the secret α used to generate the commitment $c = g^\alpha$, where g is a generator of the group. Assuming that the prover made two proofs, one with secret α_0, and the other one with secret $\alpha_1 = a\alpha_0 + b$, how can you reciver the secret witness, knowing the public values, the transcripts of the proofs, a and b?

Solution to Exercise 8.2

We then have

$$\begin{cases} g^r = g^{\alpha_0} \\ e \\ f = \alpha_0 + ex \pmod q \end{cases} \quad \text{and} \quad \begin{cases} g^{r'} = g^{\alpha_1} = g^{a\alpha_0 + b} \\ e' \\ f' = a\alpha_0 + b + e'x \pmod q \end{cases}$$

Then, we can immediately infer

$$af - f' = aex - b - e'x$$
$$x = \frac{a \cdot f - f' + b}{a \cdot e - e'}.$$

And also, $\alpha_0 = f - e \cdot x$.

8.3: Exercise 3 (Commitment scheme and batching)

Solution to Exercise 8.3

See Exercise 0 of APE9.

APE 9

9.1: Exercise 0 (Commitment scheme and batching)

By design secure public-key encryption schemes are perfectly binding commitment schemes (which are also computationally hiding, why?). Then, if perfect hiding property is not a concern, do commitment schemes really consist of a new usefull cryptographic building block? This exercise aims to build a perfectly hiding commitment scheme which supports a *batching* property that encryption schemes cannot achieve.

Let p be a prime and let $g \in QR(p)$ be an element of prime order $q > 2^l$. We let G denote the group generated by g and we let I denote the set of integers $\{1, \ldots, q\}$. Fix n random values $g_1, \ldots, g_n \in G$ and define the commitment function $\mathsf{Com}\colon I^n \mapsto G$ by

$$\mathsf{Com}(x_1, \ldots, x_n\,; r) = g^r g_1^{x_1} g_2^{x_2} \cdots g_n^{x_n}$$

1. Describe formally the commitment scheme. Discuss its efficiency and its correctness.
2. Show that the scheme is computationally binding assuming that DLog is intractable in G. That is, show that an adversary computing two openings of a commitment c for random $g, g_1, \ldots, g_n \in G$ can be used to compute discrete-log in G.

   Hint: given a pair $g, h \in G$ your goal is to find an $\alpha \in \mathbb{Z}_q$ such that $g^\alpha = h \mod p$. Choose $g_1, \ldots, g_n \in G$ so that two valid openings will reveal α.
3. Show that the scheme results in a perfectly hiding commitment on several messages. Compare the size of the construction with respect to an encryption (viewed as a commitment) of all these messages.

Solution to Exercise 9.1

1. We define $\Pi \coloneqq \langle \mathsf{Gen}, \mathsf{Com}, \mathsf{Open} \rangle$ as:
   - $\mathsf{Gen}(1^n, 1^l)$ sets pk as (p, q, g) where $q > 2^l$ ang g has order q modulo p (since $\phi(p)$ is even, that means that $g \in QR(p)$).
   - $\mathsf{Com}_{pk}(x_1, \ldots, x_n)$ provides (c, d) where:
     - $c \coloneqq g^r g_1^{x_1} g_2^{x_2} \cdots g_n^{x_n}$ (for a random $r \in \mathbb{Z}_q$)
     - $d \coloneqq (r, x_1, \ldots, x_n)$
   - $\mathsf{Open}_{pk}(c, d)$ outputs $(x_1, \ldots, x_n)$ if it can recompute c from d and pk, or $\perp$ otherwise.

   We can see that there are different possible $x_1, \ldots, x_n$ that are valid. If we fix $x_2, \ldots, x_n$, there is an x_1 such that $g_1^{x_1} = c/(g^r g_2^{x_2} \cdots g_n^{x_n})$ so there is q^n possible opening. However, it is not easy to find for a PPT algorithm.

   I should maybe have defined $d \coloneqq (r, x_1, \ldots, x_n)$ because here it is weird because Open can have different outputs.
2. For two random g, h, we need to find x such that $g^x = h$. For this task, we have access to an adversary $\mathcal{A}_\Pi$ against our scheme Π that can find (c, d_0, d_1) with different (c, d_0) and (c, d_1) that have valid openings.

   Pick a random j and set $g_i = h = g^x$ if $i = j$, or $g_i = g^{\alpha_i}$ for random α_i if $i \neq j$. We need to pick a random j so that $\mathcal{A}_\Pi$ has no idea where we've put the h (the adversary may also be adversarious against us!). From the point of view of $\mathcal{A}_\Pi$, everything looks like it should. From $c = g^r g_1^{x_1} \cdots g_n^{x_n} = g^{r'} g_1^{x_1'} \cdots g_n^{x_n'}$, and $d_0 = (r, x_1, \ldots, x_n)$ and $d_1 = (r', x_1', \ldots, _n')$ generated by

$\mathcal{A}_\Pi$, we get

$$r + \alpha_1 x_1 + \cdots + x x_j + \cdots + \alpha_n x_n \equiv r' + \alpha_1 x'_1 + \cdots + x x'_j + \cdots + \alpha_n x'_n \pmod{q}$$

so we get

$$x \equiv \frac{r' - r + \sum_{i=1 i \neq j}^{n} (x'_i - x_i)\alpha_i}{x_j - x'_j} \pmod{q}$$

We know that for at least one i^*, $x_i \neq x'_i$ so we have $\frac{1}{n}$ chance that $i^* = j$ and $x_j \not\equiv x'_j \pmod{q}$. If this is the case, we can find the inverse of $(x_j - x'_j)$ and solve the DLog problem.

3. As r is randomly selected in $\mathbb{Z}_q$ and g generates G, $g^r g_1^{x_1} \dots g_n^{x_n}$ could be any element of G, whatever the values of $(g_1, \dots, g_n)$ and $(x_1, \dots, x_n)$ so the commitment looks random and is thus perfectly hiding.

   The size is n times smaller. However, there is a difference in the "negligibility level" of the two constructs. We may thus need a bigger group for this scheme than for a standard encryption scheme in QR_p.

9.2: Exercise 1 (Jan 2011 evaluation)

The Digital Signature Standard (DSS, also often called DSA) is one of the most commonly used signature algorithms. Its three algorithms Gen, Sign and Vrfy work as follows.

- Gen: on input 1^n, select prime integers p and q such that $|q| = n$, $q|(p-1)$ and $q^2 \not| (p-1)$, together with an integer g that generates the subgroup of $\mathbb{Z}_p^*$ of prime order q. Also choose a hash function $H\colon \{0,1\}^* \mapsto \mathbb{Z}_q$. Then, select $x \leftarrow \mathbb{Z}_q$ uniformly at random, and compute $y \coloneqq g^x \mod p$. The public key is $\langle H, p, q, g, y \rangle$, and the private key is $\langle x \rangle$.
- Sign: in order to sign the message $m \in \{0,1\}^*$, choose $k \leftarrow \mathbb{Z}_q^*$ uniformly at random and set $r \coloneqq [g^k \mod p] \mod q$. Then, compute $s \coloneqq (H(m) + xr) \cdot k^{-1} \mod q$, and output the signature (r, s).
- Vrfy: compute $u_1 \coloneqq H(m) \cdot s^{-1} \mod q$ and $u_2 \coloneqq r \cdot s^{-1} \mod q$, and output 1 if and only if $r = [g^{u_1} y^{u_2} \mod p] \mod q$.

1. Show the correctness of the DSS algorithm.
2. As randomness is an expensive resource, it is proposed to select the random value k once and for all, and to sign all messages using that value of k. Is this variant of DSS secure?

   (Hint: see what you can deduce from the signature of two different messages.)

Solution to Exercise 9.2

1. We show the correctness (if the signature is well formed, it is accepted).

$$\begin{aligned}
[g^{u_1} y^{u_2} \mod p] \mod q &= [g^{H(m)\cdot s^{-1}} (g^x)^{r \cdot s^{-1}} \mod p] \mod q, \\
&= [g^{(H(m)+rx)\cdot s^{-1}} \mod p] \mod q, \\
&= [g^k \mod p] \mod q, \\
&= r.
\end{aligned}$$

2. As the same k was used for two signatures, we have the valid signatures (r, s_1) where $s_1 \coloneqq (H(m_1) + xr) \cdot k^{-1} \mod q$ and (r, s_2) where $s_1 \coloneqq (H(m_2) + xr) \cdot k^{-1} \mod q$. Now, we consider the quotient of s_1 divided by s_2 (the operations are modulo q), in order to cancel the part

depending on k, note that we can do it with two signatures only because the same k is reused:

$$\frac{s_1}{s_2} = \frac{(H(m_1) + xr) \cdot k^{-1}}{(H(m_2) + xr) \cdot k^{-1}} = \frac{H(m_1) + xr}{H(m_2) + xr}.$$

This is equivalent to: $s_1(H(m_2) + xr) = s_2(H(m_1) + xr)$, then:

$$xr = \frac{s_1 H(m_2) - s_2 H(m_1)}{s_2 - s_1}.$$

Finally, dividing by r we get x which is the secret key. As with non negligible probability (when the denominators are non null we can perform the operations described below) we can get x and sign any other message, this variant of DSS is non secure.

Alternative solution:

1. $(r, s) = ([g^k \pmod p] \pmod q, [H(m) + xr]k^{-1} \pmod q)$, then: $u_1 = H(m)s^{-1} \pmod q$, $u_2 = rs^{-1} \pmod q$, $r = [g^{u_1} g^{u_2} \pmod p] \pmod q$, $y = g^x$

$$\Rightarrow g^{u_1 + xu_2} = g^{s^{-1}(H(m)+rx)} = [g^k \pmod p] \pmod q = r$$

2. $s = (H(m) + xr)k^{-1} \pmod q$, $s' = (H(m') + xr)k^{-1} \pmod q$ ($s \neq s'$ otherwise we have a collision). $s - s' = (H(m) - H(m'))k^{-1} \pmod q$, $k = \frac{H(m) - H(m')}{s - s'}$. $s = (H(m) + xr)k^{-1}$ so $\frac{sk - H(m)}{r} = x$ where x is the secret.

9.3: Exercise 2 (RSA permutation with modulus 221)

Suppose we decide to use an RSA permutation with modulus 221, we consider RSA encryption scheme, and RSA signature.

1. What is the smallest non trivial public exponent e than can be chosen?
2. Can we choose $e = 11$? What is the corresponding private exponent d? Give the public and private key of the corresponding RSA encryption scheme.
3. Compute $c \coloneqq 219^e \pmod{221}$.
4. Verify that $c^d = 219 \pmod{221}$ as expected.
5. How Alice (owning the private key) could sign a message m? Sign the message $m = 3$ (hint: $22^7 = 61 \pmod{221}$).
6. Is 160 a valid signature for $m = 218$?

Solution to Exercise 9.3

We suggest you to use a calculator to do this exercise, the assistants said we could (pour faire taire les rageux).

1. First we have to find $\phi(221)$. As 221 is not a prime, we must find p and q such that $221 = pq$. After a few try, we find $p = 13$ and $q = 17$. Wet get then $\phi(221) = (p - 1)(q - 1) = 192$.

   The searched smallest e should respect the condition $gcd(192, e) = 1$. Again, after a few try, we find $e = 5$.

2. **Could we choose $e = 11$?**

   Yeah, since 11 is a prime, then $gcd(192, 11) = 1$.

What is the corresponding private exponent d ?

$$ed = 1 \pmod{192}$$

$$11d = 1 \pmod{1)92}$$

$$d = \frac{192k + 1}{11}$$

If we take k=2, we find $d = 35$.

Give the public and private key of the corresponding RSA encryption scheme

pk = $(221, 11)$ sk = $(221, 35)$

3.
- $219^1 = -2$
- $219^2 = 4$
- $219^4 = 16$
- $219^5 = -32$
- $219^{10} = 140$
- $219^{11} = 162$

4.
- $162^1 = -59$
- $162^2 = -55$
- $162^4 = -69$
- $162^5 = 93$
- $162^6 = 38$
- $162^7 = -32$
- $162^{35} = -2 = 219$

5. According the scheme described in the slides, $\mathsf{Sign}_{(N,d)}(m) \coloneqq [m^d mod N]$

Sign the message m = 3

- $3^1 = 3$
- $3^2 = 9$
- $3^4 = 81$
- $3^5 = 22$
- $3^{35} = 61$ (using the hint given)

6. Let's calculate (yipie)

- $218^1 = -3$
- $218^2 = 9$
- $218^4 = 81$
- $218^5 = -22$
- $218^{35} = -61 = 160$

Yes it is a valid signature

9.4: Exercise 3 (Derandomizing signatures)

Solution to Exercise 9.4

See Exercise 2 of APE10.

9.5: Exercise 4

Let f be a one-way permutation on $\{0,1\}^\lambda$. Consider the following signature scheme for messages in the set $\{1,\ldots,n\}$, where $n \in \mathsf{poly}(\lambda)$:

1. To generate keys, choose $x \leftarrow \{0,1\}^\lambda$ at random and set $y := f^n(x)$. The public key is y and the private key is x.
2. To sign message $i \in \{1,\ldots,n\}$, output $f^{n-i}(x)$ (where $f^0(x) \stackrel{\text{def}}{=} x$).
3. To verify signature σ on message i with respect to public key y, check whether $y \stackrel{?}{=} f^i(\sigma)$.

1. Show that the above is not a one-time signature scheme. Given a signature on a message i, for what messages j can an adversary output a forgery?
2. Prove that no PPT adversary given a signature on i can output a forgery on any message $j > i$ except with negligible probability.
3. Suggest how to modify the scheme so as to obtain a one-time signature scheme.
   Hint: include two values y, y' in the public key.

Solution to Exercise 9.5

1. A has $(i, \sigma(i))$ with $\sigma(i) = f^{n-i}(x)$. We know (because f is a permutation function) that:
$$f(\sigma(i)) = f^{n-i+1}(x) = f^{n-(i-1)}(x) = \sigma(i-1)$$
   Then it's possible to compute a valid forgery for every $j < i$. The scheme is then not a one time-signature.
2. Need schema drawn at TP !! It's a lot simpler with it...
   $\Pr[\text{Success}_{\mathcal{A}_\sigma}] = \epsilon(\lambda)$, $\Pr[\text{Abort}] = \frac{n-k}{n}$, $\Pr[\text{Success}] = \frac{n-k}{n-m-1}$ then:
$$\Pr[\text{Success}_{\mathcal{A}_{owf}}] = \epsilon(\lambda)\frac{n-k}{n}\frac{n-k}{n-m-1}$$
   If $\epsilon(\lambda)$ is not negligible, then the probability of success is not negligible.
3. We have $s_k = (x, x')$, $p_k = (f^n(x), f^n(x'))$. Then $m \to \sigma = (f^{n-m}(x), f^m(x'))$.

9.6: Exercise X (Jan 2011 evaluation)

Consider the following one-time signature scheme $\Pi := \langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy} \rangle$, parameterized by a PPT function $f\colon \{0,1\}^* \mapsto \{0,1\}^*$.

- Gen: on input 1^n, select $(x_0, x_1) \leftarrow \{0,1\}^n \times \{0,1\}^n$ uniformly at random, compute $(y_0, y_1) := (f(x_0), f(x_1))$ and output the pair $(pk, sk) := ((y_0, y_1), (x_0, x_1))$.
- Sign: the signature σ of the bit m is x_m.

- Vrfy: on input (m, σ), output 1 iff $y_m = f(\sigma)$.

Show that if Π is existentially unforgeable under a single-message attack, then f is a one-way function.

Solution to Exercise 9.6

Two solution have been proposed. (Actually they are the same but with a different explanation)

- Let's show that if f is not one way, then Π is not existentially unforgeable. Let $\mathcal{A}$ be the inverter of f, we will build $\mathcal{A}'$ that builds an existential forgery with non-negligible probability.
  - $\mathcal{A}'$ receives $pk = (y_0, y_1)$
  - $\mathcal{A}'$ ask the signature of 0 and gets σ, he does not really care about it
  - $\mathcal{A}'$ gives y_0 (or y_1) to $\mathcal{A}$ which outputs x_0 (or x_1) with non-negligible probability.
  - $\mathcal{A}'$ outputs $(0, x_0)$ (or $(1, x_1)$)

  Since y_1 is the image of a random x_1, we are exactly in the inverting experiment so $f(x'_1) = y_1$ with probability $\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1]$.

  We know that
  $$\Pr[\mathsf{Sig} - \mathsf{forge}^{\text{1-time}}_{\mathcal{A}',\Pi}(n) = 1] = \Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1]$$
- Let assume that f is not one way function $y = f(x)$. Then, we can recover x with a non negligible probability $\epsilon_x(n)$.

  So, $(y_0, y_1) \Rightarrow (x_0, x_1)$ with probabilities $(\epsilon_{x_0}(n), \epsilon_{x_1}(n))$. We cannot compute a pre-image by asking the oracle. So I output $(0, x_0)$ and $(1, x_1)$ as a forgery.

  The probability $\Pr[\mathsf{Sig} - \mathsf{forge}_{\mathcal{A},\Pi}(n) = 1] = \Pr[f^{-1}(\cdot)_{Inv,f(\cdot)}(n) = 1] \leq \epsilon(n)$. As Π is supposed to be existentially unforgeable under a single-message attack, then $\epsilon(n)$ is negligible and this implies that f is a one-way function.

APE 10

10.1: Exercise 1 (Authenticated encryption, or not; August Exam)

Let $\Pi := \langle \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ be an authenticated encryption scheme such that Enc encrypts messages of n bits. Do the following systems provide authenticated encryption? For those that do, briefly explain why. For those that do not, present an attack that breaks one of the security properties of an authenticated encryption scheme.

1. $\Pi' := \langle \mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}' \rangle$ with $\mathsf{Enc}'_k(m) = (\mathsf{Enc}_k(m), \mathsf{Enc}_k(m \oplus (0^{n-1} \| 1)))$ and $\mathsf{Dec}'_k(c_1, c_2) = \mathsf{Dec}_k(c_1)$ if $\mathsf{Dec}_k(c_1) \oplus \mathsf{Dec}_k(c_2) = 0^{n-1} \| 1$ and $\perp$ otherwise.
2. $\Pi' := \langle \mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}' \rangle$ with $\mathsf{Enc}'_k(m) = (\mathsf{Enc}_k(m), \mathsf{Mac}_k(m))$ and $\mathsf{Dec}'_k(c_1, c_2) = \mathsf{Dec}_k(c_1)$ if $\mathsf{Vrfy}_k(\mathsf{Dec}_k(c_1), c_2) = 1$ and $\perp$ otherwise. Here, Mac and Vrfy are deterministic algorithms that are part of a secure MAC scheme that is compatible with Gen.

Solution to Exercise 10.1

$\Pi := \langle \text{Gen, Enc, Dec} \rangle$ is an authenticated encryption scheme (AE) if it is CCA-secure and unforgeable.

1. The system Π' is not AE because it is **forgeable** and we can show it with this example. If the adversary A asks for the message m (m' corresponds to the message m with the last bit changed) to the oracle access, he will receive the cipher text (c_1, c_2), where $c_1 = \mathsf{Enc}_k(m)$ and $c_2 = \mathsf{Enc}_k(m \oplus 0^{n-1} || 1) = \mathsf{Enc}_k(m')$.

   If $\mathcal{A}$ outputs the pair (m',(c_2, c_1)), this is a forgery.

   $\mathsf{Dec}'_k(c_2, c_1) = \mathsf{Dec}_k(c_2) = \mathsf{Dec}_k(\mathsf{Enc}_k(m')) = m' \neq \perp$ because $\mathsf{Dec}_k(c_2) \oplus \mathsf{Dec}_k(c_1) = m' \oplus m = m \oplus 0^{n-1} || 1 \oplus m = 0^{n-1} || 1$. And m' has not been requested before.

   Then we have $\text{EncForge}_{A,\Pi'}(\text{n}) = 1$ and $\Pr[\text{EncForge}_{A,\Pi'}(\text{n})] = 1$. Π' is then forgeable and it is not an AE.

   With the same technique, an adversary can break the CCA-security of this scheme by querying two different messages m_1 and m_2, obtaining their encryption, sending $(m'_1, m'_2) = (m_1 \oplus 0^{n-1} || 1, m_2 \oplus 0^{n-1} || 1)$ for the challenge, and compare the encryption of m'_b with the two previously received ciphertexts.

   Note that this doesn't break CPA-security; indeed, the scheme is still CPA-secure.

2. The sytem Π' is not AE because it is not **CCA-secure** and we can show it because $Mac_k(m)$ does not assure any security (only authentication). So if We use as Mac:

   $$\mathsf{Mac}_k(m) = m || \, \mathsf{Mac}'_k(m)$$

   It is a good mac but it is trivial to show that it is not CCA-Secure. Π' is then not an AE.

   Stronger argument:

   The adversary can send two different messages m_0 and m_1 to the encryption oracle to get $(\mathsf{Enc}_k(m_0), \mathsf{Mac}_k(m_0))$ and $(\mathsf{Enc}_k(m_1), \mathsf{Mac}_k(m_1))$.

   We then output the same m_0 and m_1, and receive $(\mathsf{Enc}_k(m_b), \mathsf{Mac}_k(m_b))$.

   As Π is an athenticated encryption scheme, we know that Enc is probabilistic and secure. However, MAC is said to be deterministic, and this causes $\mathsf{Mac}_k(m_b)$ to be the same as one of the $\mathsf{Mac}_k(m_0)$, $\mathsf{Mac}_k(m_1)$ received earlier. We can thus just compare the tags, and output the corresponding b'.

The probability of success is $\Pr[b' = b] = \Pr[\mathsf{Mac}_k(m0) \neq \mathsf{Mac}_k(m_1)] = 1 - \epsilon(n)$ which is clearly well above what it should be.

10.2: Exercise 2 (Derandomizing signatures)

Let $S = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be an EUF-CMA signature scheme defined over (M, Σ), where the signing algorithm Sign is probabilistic. In particular, algorithm Sign uses randomness chosen from a space R. We let $\mathsf{S}(sk, m; r)$ denote the execution of algorithm S with randomness r. Let F be a secure PRF defined over (K, M, R). Show that the following signature scheme with deterministic signing $S' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy})$ is EUF-CMA:

$$\mathsf{G}'(1^n) \coloneqq \{(pk, sk) \leftarrow \mathcal{G}(1^n), \qquad k \leftarrow K, \qquad sk' \coloneqq (sk, k), \qquad \text{output } (pk, sk')\};$$

$$\mathsf{Sign}'((sk, k), m) \coloneqq \{r \leftarrow F_k(m), \qquad \sigma \leftarrow \mathsf{S}(sk, m; r), \qquad \text{output } \sigma\}.$$

(Hint: Define S'' which is like S' byt uses a perfect random function. Make a reduction of the security of S'' to the security of S, then build a PRF distinguisher based on a adversary against the signature. Finally, compute the link of the advantages of three relevant games.)

Solution to Exercise 10.2

We will prove this in two steps.

- The first step defines S'' so that the PRF is replaced by a true random function. We will show that in this case, if S is secure, then S'' is secure.
- The second step proves by reduction that if S'' is secure and F is a secure PRF, then S' is secure. The reduction proceeds by constructing an adversary $\mathcal{A}_{PRF}$ against the PRF, able to distinguish between the PRF and a random function, given an adversary $\mathcal{A}_{S'}$ against S'.

First, let's define S''. The only change is that

$$\mathsf{Sign}''((sk, k), m) \coloneqq \{r \coloneqq f(m), \quad \sigma \coloneqq \mathsf{S}''(sk, m), \quad \text{output } \sigma\}$$

where $\mathsf{S}''(sk, m) = \mathsf{S}(sk, m; f(m))$.

If we have an adversary against S'', we can build an adversary against S, simply by relaying oracle calls to $\mathsf{S}''_f(sk, m)$ to our oracle $\mathsf{S}(sk, m; f(m))$. As f is a random function, S sees the same randomness as with an r, so nothing changes. The probabilities are exactly the same:

$$\Pr[\mathsf{Sig} - \mathsf{forge}_{S''} = 1] = \Pr[\mathsf{Sig} - \mathsf{forge}_S = 1] = \epsilon(n)$$

for some ϵ negligible.

Now, let's do the reduction from F to S'. So, we have an adversary $\mathcal{A}_{S'}$ against S', and we build an adversary against the PRF $\mathcal{A}_F$ as follows:

1. The oracle for the PRF problem $\mathcal{O}_F$ picks $k \leftarrow \mathcal{K}$, kept secret.
2. $\mathcal{O}_F$ picks $b \leftarrow \{0, 1\}$, kept secret. The oracle defines a challenge function $g = F_k$ if $b = 1$, $g = f$ a random function if $b = 0$.
3. $\mathcal{A}_F$ runs Gen' to create a public key pk and a secret key $sk' = (sk, k)$. It discards k as it is not used by $\mathcal{A}_{S'}$ and is defined by $\mathcal{O}_F$. It keeps sk secret and sends pk to $\mathcal{A}_{S'}$.
4. When $\mathcal{A}_{S'}$ asks for $\mathsf{Sign}'((sk, k), m)$, we ask the oracle for $w = g(m)$. $w = F_k(m)$ if $b = 1$, or $w = f(m)$ with f a random function, if $b = 0$. We then run $\mathsf{S}(sk, m; r)$ and return the result to $\mathcal{A}_{S'}$.
5. When $\mathcal{A}_{S'}$ outputs (m, σ), its forgery, we outputs $b' = 1$ iff $(m, sigma)$ is a valid forgery (we can verify it using Vrfy_{pk}), $b' = 0$ otherwise.

If $b = 0$, we're playing against a true random function, and so we're actually running the scheme S''. As $\mathcal{A}_{S'}$ is not designed to handle this scheme, its security is the same as the one of S'', which is the same as the one of S.

If $b = 1$, we're playing the true game for $\mathcal{A}_{S'}$, and so its advantage $\epsilon'(n)$ is active.

Then, the difference in probabilities for the distinguishing is:

$$|\Pr[\mathcal{A}_F^{F_k}(n)] - \Pr[\mathcal{A}_F^f(n)]| = |\epsilon'(n) - \frac{1}{2}\epsilon(n)| \geq \epsilon'(n) - \epsilon(n)$$

As this difference has to be negligible, as F is a PRF, and ϵ is negligible, then ϵ' is negligible, and thus the scheme S' is secure.

10.3: Exercise 3 (Jan 11 evaluation)

Solution to Exercise 10.3

See Exercise 1 of APE9.

References

[1] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols.* CRC Press, 2 edition, 2014.