Advanced Cryptography — Final Exam
Solution

Serge Vaudenay

22.6.2017

- duration: 3h
- any document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- the exam invigilators will **not** answer any technical question during the exam
- readability and style of writing will be part of the grade

The exam grade follows a linear scale in which each question has the same weight.

1 Breaking AES Reduced to 4 Rounds

In this exercise, we consider a block cipher AES4 which is a reduced version of AES. The block cipher AES4 takes as input a key K and a plaintext block X and returns a ciphertext block Y. The key K consists of a sequence of five blocks $K_0, K_1, \ldots, K_4$. A block (like X, Y, or the K_r) is a 4×4 matrix of bytes. (A byte is a bitstring of length 8, i.e. an element of $\{0,1\}^8$.) We let $K_{r,i,j}$ be the byte at position (i,j) in K_r, $0 \leq i,j \leq 3$. The bitwise exclusive OR of blocks (bitwise, component-wise) is denoted with $\oplus$. We define AES4 as follows:

AES4(K,X):
1: $S \leftarrow X \oplus K_0$
2: **for** $r = 1$ to 4 **do**
3: $S \leftarrow \mathsf{SubBytes}(S)$
4: $S \leftarrow \mathsf{ShiftRows}(S)$
5: $S \leftarrow \mathsf{MixColumns}(S)$
6: $S \leftarrow S \oplus K_r$
7: **end for**
8: return S

The $V = \mathsf{SubBytes}(U)$ function is defined by $V_{i,j} = S(U_{i,j})$ for all (i,j), where S is a bijective operation on the set of bytes which is defined by a table. The $V = \mathsf{ShiftRows}(U)$ function is defined by $V_{i,j} = U_{i,j-i \bmod 4}$ for all (i,j). The $V = \mathsf{MixColumns}(U)$ function is defined by $V_{.,j} = \mathcal{L}(U_{.,j})$ for all j, where $U_{.,j}$ denotes the vector formed by the j-th column of U, and $\mathcal{L}$ is an invertible linear transform on the set of vectors of four bytes. (It is linear in the sense of the $\oplus$ operation.) All these functions are known by the adversary. Only K is unknown. We want to construct an adversary who will do a key recovery attack with chosen plaintexts or known plaintexts. We denote $N = 256$.

Given a block B, let $\mathcal{S}_B$ be the set of all blocks X such that $X_{i,j} = B_{i,j}$ for all (i,j) such that $i + j \bmod 4 \neq 0$. (So, only $X_{0,0}, X_{1,3}, X_{2,2}, X_{3,1}$ change.)

We also define the set $\mathcal{Z}$ of all blocks X such that $X_{i,j} = 0$ for all (i,j) such that $(i,j) \neq (0,0)$.

Q.1 Given B, we pick $X, X' \in \mathcal{S}_B$ at random. We denote by Z_r resp. Z'_r the state of encryption after round r. I.e., $Z_0 = X \oplus K_0$, $Z'_0 = X' \oplus K_0$, and

$$Z_r = \mathsf{MixColumns}(\mathsf{ShiftRows}(\mathsf{SubBytes}(Z_{r-1}))) \oplus K_r$$
$$Z'_r = \mathsf{MixColumns}(\mathsf{ShiftRows}(\mathsf{SubBytes}(Z'_{r-1}))) \oplus K_r$$

for $r = 1, \ldots, 4$. What is the probability that $Z_1 \oplus Z'_1 \in \mathcal{Z}$? We let E denote this event in what follows.

We look at the propagation of the differences between X and X'. At the beginning, differences can only occur in positions such that $i + j \bmod 4 = 0$. After the XOR to K_0, this is the same. After SubBytes*, this is the same. After* ShiftRows*, difference can only occur in positions such that $j = 0$. After* MixColumns*, differences in positions $(i, 0)$ for $i > 0$ are zero with probability $1/N^3$ because $\mathcal{L}$ is invertible. This remains so after the XOR with K_1. So, the probability is $1/N^3$.*

Q.2 If E occurs, what does $Z_2 \oplus Z'_2$ look like?

We continue to look at the propagation of the difference in $Z_{1,0,0}$. After SubBytes*, the difference can only be at position $(0,0)$. After* ShiftRows*, this is the same. After* MixColumns*, differences are only in column 0. This remains so after the XOR with K_2.*

Q.3 The set of column vectors is a vector space of dimension 32 when considered over $\mathbf{Z}_2$, and dimension 4, when considered over $\mathsf{GF}(N)$. Define four linear subspaces $\mathcal{L}_j$ of dimension 8 (over $\mathbf{Z}_2$), or 1 (over $\mathsf{GF}(N)$) such that if E occurs, then $Z_{3,.,j} \oplus Z'_{3,.,j} \in \mathcal{L}_j$ for all j.

We continue to look at the propagation of the difference in $Z_{2,.,0}$. After SubBytes*, the difference can only be at position $(.,0)$. After* ShiftRows*, the difference can only be at position (i,j) such that $i = j$. Clearly, $Z_{3,.,j} \oplus Z'_{3,.,j}$ is the result of $\mathcal{L}$ on the difference at columns j. For $j = 0$, this is $\mathcal{L}_0 = \mathcal{L}(*,0,0,0)$. For $j = 1$, this is $\mathcal{L}_1 = \mathcal{L}(0,*,0,0)$. For $j = 2$, this is $\mathcal{L}_2 = \mathcal{L}(0,0,*,0)$. For $j = 3$, this is $\mathcal{L}_3 = \mathcal{L}(0,0,0,*)$.*

Q.4 Give an algorithm which recovers a set of about N^4 possible values in which K_4 belongs to with probability $1/N^3$, with a time complexity equivalent to N^4 encryptions, and two chosen plaintexts. Explain why the attack works and justify the complexity.
HINT: recover $\mathsf{ShiftRows}^{-1}(\mathsf{MicColumns}^{-1}(K_4))$ by chunks of four bytes.

We let $\bar{K} = \mathsf{ShiftRows}^{-1}(\mathsf{MicColumns}^{-1}(K_4))$. *We note that the last three steps of encryption are equivalent to*

1: $S \leftarrow S \oplus \bar{K}$
2: $S \leftarrow \mathsf{ShiftRows}(S)$
3: $S \leftarrow \mathsf{MixColumns}(S)$

However, given the ciphertext Y*, we can invert the last two steps and get* $\bar{Y} = \mathsf{SubBytes}(Z_3) \oplus \bar{K}$ *and* $\bar{Y}' = \mathsf{SubBytes}(Z_3') \oplus \bar{K}$.
The algorithm works as follows:

1: *pick* $X, X' \in \mathcal{S}_B$ *at random*
2: *get* $Y = \mathsf{Enc}(X)$ *and* $Y' = \mathsf{Enc}(X')$ *by chosen plaintext queries*
3: *compute* $\bar{Y}$ *and* $\bar{Y}'$ *by inverting* MixColumns *and* ShiftRows *on* Y *and* Y'
4: ***for*** $j = 0$ *to* 3 ***do***
5: ***for*** *each possible* $\bar{K}_{\cdot,j}$ ***do***
6: *if* $\mathsf{SubBytes}^{-1}(\bar{Y}_{\cdot,j} \oplus \bar{K}_{\cdot,j}) \oplus \mathsf{SubBytes}^{-1}(\bar{Y}'_{\cdot,j} \oplus \bar{K}_{\cdot,j}) \in \mathcal{L}_j$*, keep this possible value for* $\bar{K}_{\cdot,j}$
7: ***end for***
8: ***end for***

The good value for $\bar{K}_{\cdot,j}$ *will be kept for sure if* E *occurs. There are* N^4 *possible values. Whether* E *occur or not, a wrong value will be kept with probability* $1/N^3$*, due to the dimension. So, we expect to keep* N *values in total. This is the case for each chunk of four bytes. Hence, we expect to collect* N^4 *possible values for* $\bar{K}$ *in total. This list contains the right* $\bar{K}$ *for sure when* E *occurs.*

Q.5 Deduce an attack to recover K_4 with good probability, using as little complexity as possible.

CHALLENGE: obtain an attack using $\sqrt{2}N^{\frac{3}{2}}$ chosen plaintexts, time complexity $\mathcal{O}(N^4)$, and memory complexity $\mathcal{O}(N^4)$.

We consider the following algorithm using q chosen plaintexts

choose B arbitrarily
pick $X_1, \ldots, X_q \in \mathcal{S}_B$ at random
get $Y_i = \mathsf{Enc}(X_i)$ chosen plaintext queries, $i = 1, \ldots, q$
initialize $4N^4$ counters $c_{j,V}$ to 0
for** each $1 \leq i < j \leq q$ **do
  get $Y = Y_i$ and $Y' = Y_j$
  compute $\bar{Y}$ and $\bar{Y}'$ by inverting MixColumns *and* ShiftRows *on Y and Y'*
  for** $j = 0$ to 3 **do
    for** $i = 0$ to 3 **do
      make the inverse table of

$$f_{i,j} : \bar{K}_{i,j} \mapsto \mathsf{SubBytes}^{-1}(\bar{Y}_{i,j} \oplus \bar{K}_{i,j}) \oplus \mathsf{SubBytes}^{-1}(\bar{Y}'_{i,j} \oplus \bar{K}_{i,j})$$

    end for
    for** each $v \in \mathcal{L}_j$ **do
      for** all $\bar{K}_{.,j}$ such that $f_{i,j}(\bar{K}_{i,j}) = v_i$ **do
        increment $c_{j,\bar{K}_{.,j}}$
      end for
    end for
  end for
end for
for** $j = 0$ to 3 **do
  let $\bar{K}_{.,j}$ be the value with maximal counter $c_{j,\bar{K}_{.,j}}$
end for

The subspace $\mathcal{L}_j$ contains N vectors. For each vector, we have on average 1 value of $\bar{K}_{.,j}$. So, the time complexity is $q + \mathcal{O}(N^4 + q^2 N)$ encryptions. The memory complexity is $4N^4$ counters.
The algorithm works like iterating $n = \binom{q}{2}$ times the previous attack. For each j, it gives a list of N possible values for $\bar{K}_{.,j}$. So, the counter of each possible value increments with probability $1/N^3$ when E does not occur. The counter of the right value always increments when E occurs, which is with probability $1/N^3$. So, it increments with probability roughly $2/N^3$. The expected value of the counters are $2n/N^3$ for the right one and n/N^3 for the wrong one. The standard deviation is roughly $\sqrt{n/N^3}$. So, for $2n/N^3 - n/N^3 \geq \sqrt{n/N^3}$, which is equivalent to $n \geq N^3$, we isolate the right value with good probability. Hence, we just take $n = N^3$ so $q = \sqrt{2}N^{\frac{3}{2}}$. The complexity becomes $\mathcal{O}(N^4)$.

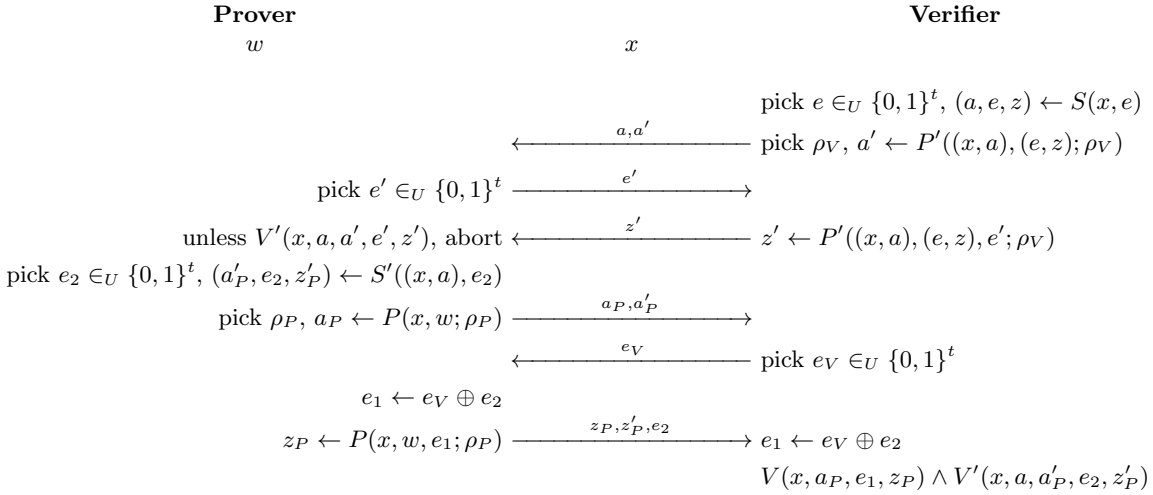
Q.6 Design an attack to recover K_4 with good probability, using $\sqrt{2}N^{\frac{15}{2}}$ known plaintexts, time complexity $\mathcal{O}(N^8)$, and memory complexity $\mathcal{O}(N^4)$.

Given $\sqrt{2}N^{\frac{15}{2}}$ random known plaintexts, we can form about N^{15} pairs. Each pair will belong to the same $\mathcal{S}_B$ structure with probability $1/N^{12}$. Hence, we can get about N^3 pairs in the same structure. Forming these pairs takes a complexity which consists of sorting the data, so less than N^8. Once this is done, we can apply the previous attack which has a lower complexity and obtain K_4.

2 ZKPoK from Sigma

This exercise is inspired from Cramer-Damgård-MacKenzie, Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions, *PKC 2000, LNCS vol. 1751, Springer.*

We consider a relation $R(x, w)$ defining a language for which we have a Σ protocol (P, V) over a challenge set $\{0,1\}^t$ with accepting predicate $V(x, a, e, z)$, Σ-simulator S, and Σ-extractor E. We define a relation $R'((x, a), (e, z))$ to hold on instance (x, a) with witness (e, z) if $V(x, a, e, z)$ is accepting. We assume that R' also has a Σ protocol (P', V') over the same challenge set $\{0,1\}^t$ with accepting predicate $V'(x, a, a', e', z')$, Σ-simulator S', and Σ-extractor E'. We consider the following protocol:



Q.1 In the first part of the protocol, recognize and isolate a commitment on the value e and a proof of knowledge of a valid opening of this commitment. Fully describe the commitment scheme. Fully describe the proof of knowledge.

In the first step of the protocol, the verifier commits to some e without revealing it by using the conversion from a Σ-protocol to a commitment scheme. For that, he runs S with challenge e and sends the commit value a. He proves in a Σ protocol that he knows a valid (e, z), so that he knows how to open the commitment.
The commitment scheme was seen in the course. To commit on e, the sender runs $S(x, e) \to (a, e, z)$ and uses a as a commit value and (e, z) as the opening value. To open a commitment a with (e, z), we just check $V(x, a, e, z)$, i.e. the relation $R'((x, a), (e, z))$. Hence, the (P', V') protocol is a proof of knowledge of a valid opening of the commitment a.

Q.2 In the second part of the protocol, recognize a proof of knowledge of either w for $(R(x, w)$ or (e, z) for $R'((x, a), (e, z))$.

The second part of the protocol is a standard OR proof for R and R'. The prover proves that he knows either w for R or (e, z) for R'. He can as he actually knows w. The OR proof runs in parallel the Σ protocols for R and R' but the challenges are chosen by the prover with a XOR e_V imposed by the verifier. If we know one of the two witnesses, we can run the corresponding Σ protocol with any challenge. If we do not, we can anticipate the challenge and use the simulator of the corresponding Σ protocol. If we know one of the two challenges, we anticipate a random challenge for the ignored witness and we select the challenge for the known witness with the correct XOR.
This OR proof was the subject of a previous exam.

Q.3 Show that the protocol is complete and runs in polynomial time $\mathsf{poly}(t, |x|)$ (where $|x|$ is the length of x) for the verifier.

All protocols run by the verifier are PPT algorithm. So, the protocol runs in polynomial time for the verifier.
The first part of the protocol is complete thanks to the second Σ protocol. It is a proof that the verifier knows (e, z), a valid opening of the commitment.
The second part of the proof is an OR-proof. It completes thanks to the simulator S' and the completeness of the Σ protocol for R.

Q.4 Show that the protocol is zero-knowledge by constructing a black-box simulator.

In the first part of the protocol, we simulate a prover normally to the verifier. Then, we rewind the verifier and run the simulation again. As it is likely to produce a different challenge e', we can use the extractor E' to extract a valid $(\bar{e}, \bar{z})$ witness for R'. Then, we can simulate a prover who knows $(\bar{e}, \bar{z})$ in the OR proof: we pick e_1, run $(a_P, e_1, z_P) \leftarrow S(x, e_1)$, $a'_P \leftarrow P'(x, a, \bar{e}, \bar{z}; \rho_P)$, send (a_P, a'_P), get e_V, set $e_2 = e_V \oplus e_1$, run $z'_P \leftarrow P'(x, a, \bar{e}, \bar{z}, e_e; \rho_P)$, and send (z_P, z'_P, e_2). We can easily see that the distribution is correct.

Q.5 Construct a knowledge extractor for this protocol to prove that it is a zero-knowledge proof of knowledge for R.

The extractor runs the prover twice with the same random coins and simulate the verifier the same way in both executions, except for issuing e_V where they may fork. Like in the proof which was seen in the course, we obtain two executions with the same transcript until e_V then two transcripts $e_V^1, z_P^1, {z'}_P^1, e_2^1$ and $e_V^2, z_P^2, {z'}_P^2, e_2^2$. We assume $e_V^1 \neq e_V^2$.
Let $e_1^i = e_V^i \oplus e_2^i$. If $e_1^1 \neq e_1^2$, we have two transcripts a_P, e_1^1, z_P^1 and a_P, e_1^2, z_P^2 with different challenges so E extracts a witness w for R.
If now $e_1^1 = e_1^2$, we must have $e_2^1 \neq e_2^2$, so we get two transcripts $a'_P, e_2^1, {z'}_P^1$ and $a'_P, e_2^2, {z'}_P^2$ with different challenges so E' extracts a witness $(\bar{e}, \bar{z})$ for R'. Since the extractor simulated the verifier, he already has some witness (e, z) for R'. If $e \neq \bar{e}$, we can use E again to obtain a witness w for R.
What remains to show is that $e = \bar{e}$ occurs with probability 2^{-t}. This is due to the extractor revealing no information about e to the prover so that $\bar{e}$ and e are statistically independent. As e is uniformly distributed, this concludes the proof.

3 PRP versus Left-or-Right

WARNING: in this exercise, the definitions which are proposed are not correct. Instead of saying there is a negligible function which major the advantage of any *adversary, we should have said any adversary has a negligible advantage.*

Given a security parameter (which is implicit and omitted from notations for better readability), we consider a pair $(\mathsf{Enc}, \mathsf{Dec})$ of functions from $\{0,1\}^k \times \{0,1\}^n$ to $\{0,1\}^n$ (k and n are functions of the security parameter). These functions are such that for all K and X, we have

$$\mathsf{Dec}(K, \mathsf{Enc}(K, X)) = X$$

It is assumed that there are implementations which can evaluate both functions in polynomial time complexity (in terms of the security parameter). We define several security notions.

PRP. We say that this pair is a *pseudorandom permutation* (PRP) if there exists a negligible function negl such that for all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, we have $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 1) \to 1] \leq \mathsf{negl}$, where $\Gamma^{\mathsf{PRP}}(\mathcal{A}, b)$ is the PRP game defined as follows:

$\Gamma^{\mathsf{PRP}}(\mathcal{A}, b)$:
1: initialize a list $\mathcal{L}$ to empty
2: pick $K \in \{0,1\}^k$ uniformly at random
3: pick a permutation Π over $\{0,1\}^n$ uniformly at random
4: run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
5: return b'

subroutine $\mathcal{O}(x)$:
6: if $x \in \mathcal{L}$ abort
7: insert x in $\mathcal{L}$
8: **if** $b = 0$ **then**
9: return $\mathsf{Enc}(K, x)$
10: **else**
11: return $\Pi(x)$
12: **end if**

LoR. We say that this pair is *LoR-secure* if there exists a negligible function negl such that for all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, we have $\Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 1) \to 1] \leq \mathsf{negl}$, where $\Gamma^{\mathsf{LoR}}(\mathcal{A}, b)$ is the left-or-right game defined as follows:

$\Gamma^{\mathsf{LoR}}(\mathcal{A}, b)$:
1: initialize two lists $\mathcal{L}_l$ and $\mathcal{L}_r$ to empty
2: pick $K \in \{0,1\}^k$ uniformly at random
3: run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
4: return b'

subroutine $\mathcal{O}(x_l, x_r)$:
5: if $x_l \in \mathcal{L}_l$ or $x_r \in \mathcal{L}_r$, abort
6: insert x_l in $\mathcal{L}_l$ and x_r in $\mathcal{L}_r$
7: **if** $b = 0$ **then**
8: return $\mathsf{Enc}(K, x_l)$
9: **else**
10: return $\mathsf{Enc}(K, x_r)$
11: **end if**

We want to show the equivalence between these notions.

Q.1 Is the list management important in each security definition (or: what happens with modified definitions in which we remove the lists)? Justify your answer.

We consider the games $\Gamma^{\mathsf{PRP}*}$ and $\Gamma^{\mathsf{LoR}*}$ which are the same as Γ^{PRP} and Γ^{LoR}, respectively, without any list management or abort.
The list management is not important in the PRP security. Indeed, we could simulate a PRP$*$ *adversary $\mathcal{A}$ repeating queries by a* PRP *adversary $\mathcal{B}$ who does not repeat them, by simulating $\mathcal{A}$, remembering his queries and the responses, and simulating repeating queries instead of querying them. We would have* $\Pr[\Gamma^{\mathsf{PRP}*}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}, b) \to 1]$.
The list management is important in the LoR security. Indeed, the following adversary outputs b with probability 1 in a LoR$*$ *game in which repetitions are allowed:*
$\mathcal{A}^{\mathcal{O}}$*:*
1: *pick $x, y \in \{0,1\}^n$ such that $x \neq y$*
2: *query $u = \mathcal{O}(x, y)$*
3: *query $v = \mathcal{O}(y, y)$*
4: *answer $1_{u=v}$*
So, $\Pr[\Gamma^{\mathsf{LoR}*}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{LoR}*}(\mathcal{A}, 1) \to 1] = 1$. *This is not negligible. So, no* LoR$*$ *security is feasible. Nevertheless, we will see that LoR and PRP are equivalent.*

Q.2 We consider the following hybrid game:
$\Gamma^{\mathsf{hyb}}(\mathcal{A}, b)$:
1: initialize a list $\mathcal{L}$ to empty
2: pick $K \in \{0,1\}^k$ uniformly at random
3: pick a permutation Π over $\{0,1\}^n$ uniformly at random
4: run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
5: return b'
subroutine $\mathcal{O}(x)$:
6: if $x \in \mathcal{L}$ abort
7: insert x in $\mathcal{L}$
8: **if** $b = 0$ **then**

9: return $\mathsf{Enc}(K, x)$
10: **else**
11: return $\mathsf{Enc}(K, \Pi(x))$
12: **end if**

Show that for all $\mathcal{A}$ playing the PRP game and any b, we have $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{hyb}}(\mathcal{A}, b) \to 1]$.

For $b = 0$, the result is obvious: by getting rid of steps which are never executed, we can see that the two games are the same. So, we concentrate on $b = 1$.
If K is random and Π is an independent uniformly distributed permutation, then $\Pi'(x) = \mathsf{Enc}(K, \Pi(x))$ is also an independent uniformly distributed permutation. So, a bridging step in which we replace $\mathsf{Enc}(K, \Pi(x))$ by $\Pi'(x)$ with Π' selected randomly produces the same result.

Q.3 Given $\mathcal{A}$ playing the PRP game, we define $\mathcal{B}$ playing the LoR game as follows:
$\mathcal{B}^{\mathcal{O}}$:
1: pick a permutation Π over $\{0,1\}^n$ uniformly at random
2: run $\mathcal{A}$
when $\mathcal{A}$ makes a query x to its oracle, answer by $\mathcal{O}(x, \Pi(x))$
3: return the same output as $\mathcal{A}$

Show that $\Pr[\Gamma^{\mathsf{hyb}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}, b) \to 1]$ for any b.

First of all, it is clear that some x repeats if and only if some $\Pi(x)$ repeats, because Π is a permutation. So, removing the $\mathcal{L}_r$ management in the LoR game with $\mathcal{B}$ does not change the outcome of the game. Then, the LoR game without $\mathcal{L}_r$ management can be changed into the hybrid game by bridging steps. So, $\Pr[\Gamma^{\mathsf{hyb}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}, b) \to 1]$.

Q.4 Deduce that LoR-security implies PRP.
CAUTION: adversaries must be PPT.

The adversary $\mathcal{B}$ in the previous question is not polynomially bounded as it must pick a random Π. However, we can perfectly simulate it by using the lazy sampling technique: $\mathcal{B}'$ keeps a table of $(x, \Pi(x))$ pairs which is initially empty and, upon a new query x, checks if it is in the table, and if not, picks a random output y which is different than all previous ones, then insert (x, y) in the table.
If we have LoR security, given a (PPT) PRP adversary $\mathcal{A}$, the previous reduction makes a PPT adversary $\mathcal{B}'$ playing the LoR game and such that $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}', b) \to 1]$ for any b. Since $\Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}', 0) \to 1] - \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{B}', 1) \to 1] \leq \mathsf{negl}$, we have $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{A}, 1) \to 1] \leq \mathsf{negl}$. Since this holds for any PPT $\mathcal{A}$, we obtain PRP security.

Q.5 Using the following game, show that PRP security implies LoR security. Give a precise proof with the reductions.

$\Gamma^{\mathsf{generic}}(\mathcal{A}, b, c)$:

1: initialize two lists $\mathcal{L}_l$ and $\mathcal{L}_r$ to empty
2: pick $K \in \{0,1\}^k$ uniformly at random
3: pick a permutation Π over $\{0,1\}^n$ uniformly at random
4: run $b' \leftarrow \mathcal{A}^{\mathcal{O}}$
5: return b'

subroutine $\mathcal{O}(x_l, x_r)$:

6: if $x_l \in \mathcal{L}_l$ or $x_r \in \mathcal{L}_r$, abort
7: insert x_l in $\mathcal{L}_l$ and x_r in $\mathcal{L}_r$
8: **if** $b = 0$ **then**
9: **if** $c = 0$ **then**
10: return $\mathsf{Enc}(K, x_l)$
11: **else**
12: return $\Pi(x_l)$
13: **end if**
14: **else**
15: **if** $c = 0$ **then**
16: return $\mathsf{Enc}(K, x_r)$
17: **else**
18: return $\Pi(x_r)$
19: **end if**
20: **end if**

Let $\mathcal{A}$ be any (PPT) LoR adversary.
We have $\Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, b) \to 1] = \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, b, 0) \to 1]$.
When $c = 1$, as the queries never repeat, the oracle always returns a random answer which is different from all previous ones, no matter the value of b. So, we have $\Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 0, 1) \to 1] = \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 1, 1) \to 1]$.
Using a bridging step, we construct $\mathcal{B}_b$ such that for all b and c, $\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_b, c) = \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, b, c) \to 1]$. *So*

$$
\begin{aligned}
& \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 0) \to 1] - \Pr[\Gamma^{\mathsf{LoR}}(\mathcal{A}, 1) \to 1] \\
&= \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 0, 0) \to 1] - \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 1, 0) \to 1] \\
&= \big(\Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 0, 0) \to 1] - \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 0, 1) \to 1]\big) - \\
&\quad \big(\Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 1, 0) \to 1] - \Pr[\Gamma^{\mathsf{generic}}(\mathcal{A}, 1, 1) \to 1]\big) \\
&= \big(\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_0, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_0, 1) \to 1]\big) - \\
&\quad \big(\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_1, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_1, 1) \to 1]\big) \\
&= \big(\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_0, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}_0, 1) \to 1]\big) + \\
&\quad \big(\Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}'_1, 0) \to 1] - \Pr[\Gamma^{\mathsf{PRP}}(\mathcal{B}'_1, 1) \to 1]\big) \\
&\leq 2\mathsf{negl}
\end{aligned}
$$

where $\mathcal{B}'_b$ gives the opposite answer to $\mathcal{B}_b$. As $2\mathsf{negl}$ *is a negligible function, this is negligible. This applies to any $\mathcal{A}$. Hence, we have LoR security.*