Homework 1

solutions by Luigi Russo, 1699981

1 Perfect Secrecy

(a) Prove or refute. An encryption scheme $(\texttt{Enc}, \texttt{Dec})$ with key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is perfectly secret if and only if the following holds: for every probability distribution M over $\mathcal{M}$, and every $c_0, c_1 \in \mathcal{C}$, we have $\mathbb{P}[C = c_0] = \mathbb{P}[C = c_1]$, where $C := \texttt{Enc}(K, M)$ with K uniform over $\mathcal{K}$.

SOL I am going to refute the previous claim. Indeed, it is easy to show that it is possible to have a non-uniform ciphertext distribution, still preserving the perfect secrecy property. Consider the following scheme Π:

$$\texttt{Enc}(m) = (m \oplus r)||b = c'||b$$
$$\texttt{Dec}(c) = \texttt{Dec}(c'||b) = c' \oplus r$$

with $r \leftarrow_{\$} \{0,1\}^{\lambda}$ and $b \in \{0,1\}$, such that $\mathbb{P}[b = 0] > \mathbb{P}[b = 1]$[1] > 0.

Note that this is a simple variant of OTP, except that we append a bit b to the ciphertext: this scheme is clearly correct, since $\texttt{Dec}$ algorithm simply ignores the last bit of the ciphertext and then applies the xor with the secret key. Note that the choice of b is completely independent of the message. So we can claim this scheme is perfectly secure because:

$$\begin{aligned} &\mathbb{P}[M = m | C = c] = \\ &\text{(b is chosen independently of m)} \rightarrow = \mathbb{P}[M = m | C = c'||\cdot] \\ &\text{(OTP)} \rightarrow = \mathbb{P}[M = m] \end{aligned}$$

However, the distribution over $\mathcal{C}$ is not uniform: indeed, let's consider the case for $\mathbb{P}[b = 0] = \frac{2}{3}$. The ciphertexts ending with a 0 bit are two times as likely as ciphertexts ending with bit 1.

□

(b) Let $(\texttt{Enc}, \texttt{Dec})$ be a perfectly secret encryption scheme over message space $\mathcal{M}$ and key space $\mathcal{K}$, satisfying the following relaxed correctness requirement: there exists $t \in \mathbb{N}$ such that, $\forall m \in \mathcal{M}$, it holds that $\mathbb{P}[\texttt{Dec}(k, \texttt{Enc}(k, m)) = m] \geq 2^{-t}$ (where the probability is over the choice of $k \leftarrow_{\$} \mathcal{K}$). Prove that $|\mathcal{K}| \geq |\mathcal{M}| \cdot 2^{-t}$.

SOL Assume not: so $|\mathcal{K}| < 2^{-t}|\mathcal{M}|$. Let M be uniform over $\mathcal{M}, c \in \mathcal{C} : \mathbb{P}[C = c] > 0$ and define $\mathcal{M}' = \{m \in \mathcal{M} : \mathbb{P}[m = \texttt{Dec}(k, c)] \geq 2^{-t}\}_{k \in \mathcal{K}}$.

[1] it's crucial that this probability is nonzero, otherwise $\mathcal{C}$ does not contain messages ending in 1.

We have that, $\forall k \in \mathcal{K}$:

$$\sum_{m \in \mathcal{M}'} \mathbb{P}[m = \texttt{Dec}(k, c)] \leq 1 \tag{1}$$

$$\sum_{m \in \mathcal{M}'} \mathbb{P}[m = \texttt{Dec}(k, c)] \geq 2^{-t} \cdot |\{m : \texttt{Dec}(k, c) = m\}| \tag{2}$$

This implies that:

$$2^{-t} \cdot |\{m : \texttt{Dec}(k, c) = m\}| \leq 1$$

If we now sum over the key-space $\mathcal{K}$ we have that:

$$2^{-t} \cdot |\mathcal{M}'| \leq 2^{-t} \cdot \sum_{k \in \mathcal{K}} |\{m : \texttt{Dec}(k, c) = m\}| \leq |\mathcal{K}| < 2^{-t} \cdot |\mathcal{M}|$$

from which we derive that $|\mathcal{M}| < |\mathcal{M}'|$, i.e. $\exists m^* \in \mathcal{M} \backslash \mathcal{M}'$. Note that $\forall k \in \mathcal{K}, \texttt{Enc}(k, m^*) \neq c$, otherwise we would have that

$$\mathbb{P}[\texttt{Dec}(k, c) = m^*] < 2^{-t} \qquad (m^* \in \mathcal{M} \backslash \mathcal{M}') \tag{3}$$

$$\mathbb{P}[\texttt{Dec}(k, c) = m^*] \geq 2^{-t} \qquad \text{(relaxed correctness)} \tag{4}$$

that is clearly an absurd. But now we have that $\mathbb{P}[M = m^*] = \frac{1}{|\mathcal{M}|}$ (it is uniform by assumption). However, $\mathbb{P}[M = m^* | C = c] = 0 \neq \frac{1}{|\mathcal{M}|}$ and this violates the perfect secrecy of the scheme. $\square$

2 Universal Hashing

(a) A family $\mathcal{H} := \{h_s : X \to Y\}_{s \in \mathcal{S}}$ of hash functions is called t-wise independent if for all sequences of distinct inputs $x_1, \ldots, x_t \in \mathcal{X}$, and for any output sequence $y_1, \ldots, y_t \in \mathcal{Y}$ (not necessarily distinct), we have that:

$$\mathbb{P}[h_s(x_1) = y_1 \wedge \ldots \wedge h_s(x_t) = y_t, s \leftarrow_{\$} S] = \frac{1}{|\mathcal{Y}|^t}$$

(i) For any $t \geq 2$, show that if $\mathcal{H}$ is t-wise independent, then it is also $(t-1)$-wise independent.
SOL In order to be $(t-1)$-wise independent, we have to prove that:

$$\mathbb{P}[h_s(x_1) = y_1 \wedge \ldots \wedge h_s(x_{t-1}) = y_{t-1}, s \leftarrow_{\$} S] = \frac{1}{|\mathcal{Y}|^{t-1}}$$

We can easily see that:

$$\begin{aligned}
&\mathbb{P}[\wedge_{i=1}^{t-1}(h_s(x_i) = y_i), s \leftarrow_{\$} S] = \\
&\text{(law of total probability)} \rightarrow = \sum_{y_t \in \mathcal{Y}} \mathbb{P}[\wedge_{i=1}^{t}(h_s(x_i) = y_i), s \leftarrow_{\$} S] \\
&\text{(by definition of } \mathcal{H}) \rightarrow = \sum_{y_t \in \mathcal{Y}} \frac{1}{|\mathcal{Y}|^t} \\
&= \frac{1}{|\mathcal{Y}|^{t-1}}
\end{aligned}$$

that concludes the proof. $\square$

(ii) Let q be a prime. Show that the family $\mathcal{H} = \{h_s : \mathbb{Z}_q \to \mathbb{Z}_q\}_{s \in \mathbb{Z}_q^3}$ defined by

$$h_s(x) \coloneqq h_{s_0,s_1,s_2}(x) \coloneqq s_0 + s_1 \cdot x + s_2 \cdot x^2 \bmod q$$

is 3-wise independent.

SOL By applying the definition, we should prove that, for a uniformly random choice of s:

$$\mathbb{P}[h_s(x_1) = y_1 \wedge h_s(x_2) = y_2 \wedge h_s(x_3) = y_3] = \frac{1}{q^3}$$

We have that:

$$\begin{aligned}
\mathbb{P}[\wedge_{i=1}^{3}(h_s(x_i) = y_i)] = & \\
&= \mathbb{P}\left[\begin{cases} s_2 \cdot x_1^2 + s_1 \cdot x_1 + s_0 = y_1 \\ s_2 \cdot x_2^2 + s_1 \cdot x_2 + s_0 = y_2 \\ s_2 \cdot x_3^2 + s_1 \cdot x_3 + s_0 = y_3 \end{cases}\right] \\
&= \mathbb{P}\left[\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}\begin{pmatrix} s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right] \\
&= \mathbb{P}\left[\begin{pmatrix} s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}^{-1}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right] \\
\text{(for some constant k)} \to &= \mathbb{P}\left[\begin{pmatrix} s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix}\right] \\
(s \text{ is uniformly random}) \to &= \frac{1}{q^3}
\end{aligned}$$

Note that this is valid if and only if we can invert that matrix, i.e. its determinant has to be nonzero.

$$\begin{aligned}
\det\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix} &= x_1^2x_2 + x_1x_3^2 + x_2^2x_3 - x_2x_3^2 - x_1^2x_3 - x_1x_2^2 \\
&= (x_1 - x_2)(x_1 - x_3)(x_2 - x_3)
\end{aligned}$$

(inputs are distinct by hypothesis) $\to \neq 0$

□

(b) Say that X is a (k, n)-source if $X \in \{0,1\}^n$, and the min-entropy of X is at least k. Answer the following questions:

(i) Suppose that $l = 128$; what is the minimal amount of min-entropy needed in order to obtain statistical error $\epsilon = 2^{-80}$ when applying the leftover hash lemma? What is the entropy loss?

SOL Leftover hash lemma states that, given pairwise independent $\mathcal{H} = \{h_s : \{0,1\}^n \to \{0,1\}^l\}_{s \in \{0,1\}^d}, h_S(X)$ is a (k, ϵ)-extractor for a value of min-entropy $k \geq l + \delta$, where $\delta = 2\log(1/\epsilon) - 2$ is the entropy loss.

In this case we want to compute k^*, i.e. the minimum k satisfying the inequality. So we have:

$$\delta = 2\log(1/2^{-80}) - 2 = 2\log(2^{80}) - 2 = 2 \cdot 80 - 2 = 158$$
$$\Rightarrow k \geq 128 + 158 = 286 = k^*$$

□

(ii) Suppose that $k = 238$; what is the maximal amount of uniform randomness that you can obtain with statistical error $\epsilon = 2^{-80}$ when applying the leftover hash lemma? Explain how to obtain $l = 320$ using computational assumptions.
SOL We follow a similar approach, this time we are interested in l^*, i.e. the maximum l satisfying the inequality:

$$\delta = 158 \leftarrow \text{(same as before)}$$
$$238 \geq l + 158 \Rightarrow l \leq 238 - 158 = 80 = l^*$$

Now, in order to obtain 320 bits of output, we can make use of a PRG $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{4\lambda}$, for $\lambda = 80$. There is a well known theorem that guarantees us the existence of such G (since the stretch is polynomial in λ) under the assumption that OWF exist. In this way, however, it should be noticed that we are moving from statistical equivalence between distributions (thanks to random extraction) to computational equivalence (we have deeply discussed the differences and the practical implications of this hypothesis). □

3 One-way Functions

(a) Let $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$ be a PRG with λ-bit stretch. Prove that G is by itself a one-way function.
SOL

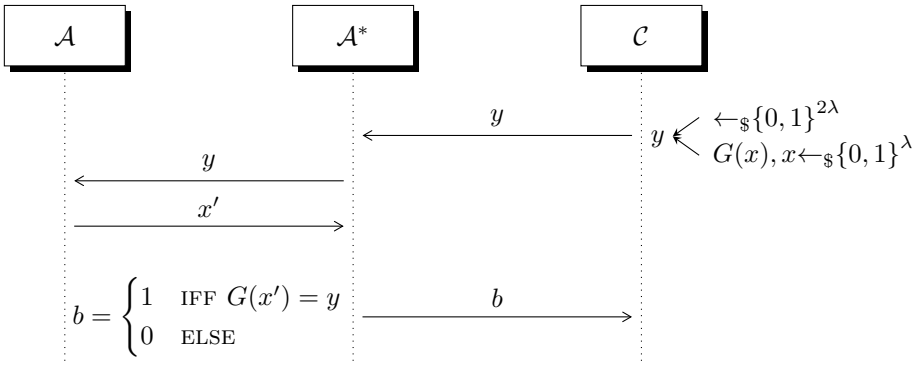

Figure 1: Base an attack to PRG G, assuming the existence of a PPT attacker $\mathcal{A}$, able to find pre-images of G with non negligible probability. $\mathcal{A}^*$ returns a bit b to the challenger $\mathcal{C}$, in order to distinguish.

Assume G is not a OWF: then there exists some PPT attacker $\mathcal{A}$ able to find a pre-image x' of input values y (see figure 1). Let us define valid[2] an input y if $\exists x : G(x) = y$.

[2]just a shortcut to indicate that $y \in G(\cdot)$.

We have that $\mathbb{P}[\mathcal{A}(y) = 1 : x \leftarrow_{\$} \{0,1\}^{\lambda}; y = G(x)] \geq \frac{1}{p(\lambda)}$, for some $p(\lambda) \in poly(\lambda)$ for hypothesis. Moreover, the probability that a random 2λ bits string is valid is at most equal to the ratio between the number of valid inputs and the number of possible inputs: this ratio is equal to $\frac{2^{\lambda}}{2^{2\lambda}} = 2^{-\lambda}$. From which we derive that $\mathbb{P}[\mathcal{A}(y) = 1 : y \leftarrow_{\$} \{0,1\}^{2\lambda}] \leq 2^{-\lambda}$. So we have:

$$\mathbb{P}[\mathcal{A}(y) = 1 : x \leftarrow_{\$} \{0,1\}^{\lambda}; y = G(x)] - \mathbb{P}[\mathcal{A}(y) = 1 : y \leftarrow_{\$} \{0,1\}^{2\lambda}] \geq \frac{1}{p(\lambda)} - 2^{-\lambda} \notin negl(\lambda)$$

□

(b) Let $f : \{0,1\}^{n} \to \{0,1\}^{n}$ be a OWF. Consider the function $g : \{0,1\}^{n+\log n} \to \{0,1\}^{n+\log n+1}$ defined by $g(x||j) := (f(x), j, x_j)$, where $x := (x_1, \ldots, x_n)$, and j is interpreted as an integer in $[n]$ (i.e. $|j| = \log n$).

(i) Show that g is a OWF if f is.
SOL Assume not. Then there exists a PPT attacker $\mathcal{A}$ able to invert g with non negligible probability: if so, we show that we can break f too (see figure 2).

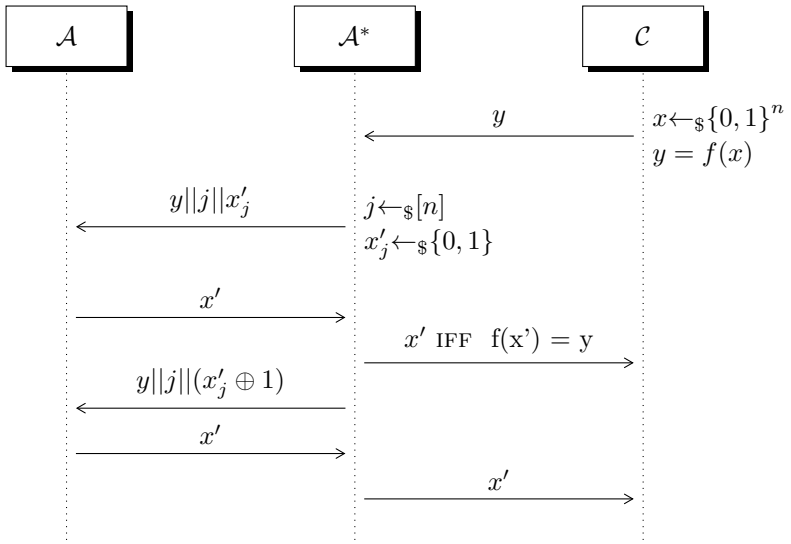

Figure 2: Base an attack to OWF f, assuming the existence of a PPT attacker $\mathcal{A}$, able to invert g with non negligible probability. $\mathcal{A}^*$ wins the game if it finds a value x' such that $f(x') = y$ provided by the challenger $\mathcal{C}$.

Our attacker tries to guess the value for the j-th bit and queries $\mathcal{A}$ to find the pre-image of $y||j|x_j$. However, it could be that the guess is wrong: but we can try again, flipping the j-th bit, if we realize that $f(x') \neq y$. Then, it is obvious that the probability to win the game, i.e. to find some valid pre-image of y is exactly equal to the probability of finding a pre-image of a valid y^* for attacker $\mathcal{A}$: and this is non negligible by assumption. □

(ii) Show that for every $i \in [n']$ there is a PPT algorithm A_i for which

$$\mathbb{P}[A_i(g(x')) = x'_i : x' \leftarrow_{\$} \{0,1\}^{n+\log n}] \geq \frac{1}{2} + \frac{1}{2n}$$

where $x' = (x_1, \ldots, x_{n'})$

SOL Let $y \in \{0,1\}^{n+\log n}, y' \in \{0,1\}^n, j \in \{0,1\}^{\log n}$.

We define $A_i(y) = A_i(y'||j||\cdot) = \begin{cases} y_i & \text{IF } i > n \vee i = j \\ 0 & \text{ELSE} \end{cases}$

Then we can prove that $\mathbb{P}[A_i(g(x')) = x'_i : x' \leftarrow_\$ \{0,1\}^{n+\log n}] \geq \frac{1}{2} + \frac{1}{2n}$

First of all, A_i is a PPT $\forall i$, since it only parses the input string, whose length is $n+\log n+1$. As for the bound, for all indexes $i > n$ (i.e. the last $\log n$ bits) A_i finds the correct value, because it is embedded in the input. For all other indexes, there are two cases: if $i = j$ (this happens with probability $= 1/n$) A_i returns the correct value with probability 1 (again, the value is visible to the attacker); if $i \neq j$ (with probability $\frac{n-1}{n}$) A_i can at least guess (returning 0) and win with probability $\frac{1}{2}$. If we take the weighted average we prove the bound also for $i \leq n$, since:

$$\forall i \leq n,\ \mathbb{P}[A_i(g(x')) = x'_i : x' \leftarrow_\$ \{0,1\}^{n+\log n}] \geq \frac{1}{n} + \frac{n-1}{2n} = \frac{1}{2} + \frac{1}{2n}$$

□

4 Pseudo-random Generators

(a) Let $G_1, G_2 : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$ be two deterministic functions mapping λ bits into $\lambda + l$ bits (for $l \geq 1$). You know that at least one of G_1, G_2 is a secure PRG, but you don't know which one. Show how to design a secure PRG $G^* : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda+l}$ by combining G_1 and G_2.

SOL I claim that $G^* := G_1(s_1) \oplus G_2(s_2), s_1, s_2 \leftarrow_\$ \{0,1\}^\lambda$ is a PRG. Indeed, assume without loss of generality that G_1 is a PRG and assume that there exists a PPT attacker $\mathcal{A}$ able to distinguish, with non negligible probability, between $G^*(U_\lambda||U_\lambda)$ and $U_{\lambda+l}$. Then we can build an attacker $\mathcal{A}^*$ that breaks G_1 (see figure 3).

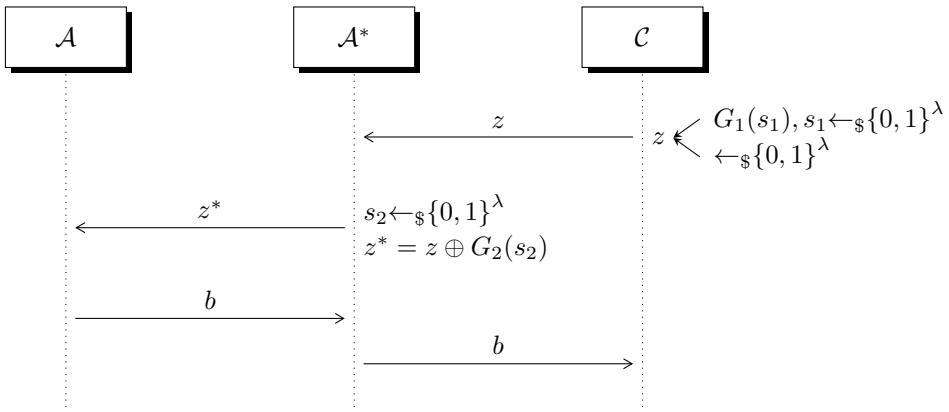

Figure 3: Base an attack to PRG G_1, assuming the existence of a PPT attacker $\mathcal{A}$, able to break the security of G^* with non negligible probability. $\mathcal{A}^*$ returns a bit b to the challenger $\mathcal{C}$, in order to distinguish.

Note that is crucial that s_1, s_2 are both uniformly chosen at random, otherwise G^* is not guaranteed to be a secure PRG. □

(b) Can you prove that your construction works when using the same seed $s^* \in \{0,1\}^\lambda$ for both G_1 and G_2? Motivate your answer.

SOL A simple counterexample is for $G_2 := G_1 \oplus a$, for some $a \in \{0,1\}^{\lambda+l}$. Then $G^*(s^*, s^*) = a, \forall s^*$. Note, however, that if the two seeds are both chosen at random, this happens only with negligible probability, so we can tolerate the existence of such a bad case. □

5 Pseudo-random Functions

(a) Show that no PRF family can be secure against computationally unbounded distinguishers.

SOL Let us consider any PRF family $F_k : \{0,1\}^n \to \{0,1\}^l, k \in \{0,1\}^\lambda, n, l \in poly(\lambda)$. An unbounded attacker could "simply" brute-force[3] $F_k(\cdot), \forall k$. At this point he queries the challenger, with all possible inputs x_i, receiving back y_i and returns 1 iff $\exists k^* : \forall i, F_{k^*}(x_i) = y_i$. Of course, if the challenger returns values computed with the PRF, the distinguisher will return 1 with probability 1. Moreover, we know that the number of possible distinct functions associated with F is at most 2^λ (since they depend on the key k); on the contrary, the number of possible functions from $2^n \to 2^l$ is $(2^l)^{2^n}$. The probability that the challenger, selecting at random a function R, picks a function that is equivalent to one of the functions F_k is $\leq \frac{|K|}{|\mathcal{R}(n,l)|} = \frac{2^\lambda}{(2^l)^{2^n}}$.

This means that

$$\mathbb{P}[\mathcal{A}(y) = 1 : y = F_k(x); k \leftarrow_\$ \{0,1\}^l] - \mathbb{P}[\mathcal{A}(y) = 1 : y = R(x); R \leftarrow_\$ \mathcal{R}(n,l)] \geq 1 - \frac{2^\lambda}{(2^l)^{2^n}}$$

that is non negligible, since I assume that $(2^l)^{2^n} >> 2^\lambda$. □

(b) Analyze the following candidate PRFs. For each of them, specify whether you think the derived construction is secure or not; in the first case prove your answer, in the second case exhibit a concrete counterexample.

(i) $F_k(x) = G'(k) \oplus x$, where $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$ is a PRG, and G' denotes the output of G truncated to λ bits.

SOL This solution does not work, since we have that, for two inputs x_1, x_2:

$$x_1 \oplus x_2 = F_k(x_1) \oplus F_k(x_2), \forall k$$

Observed that, we can build an efficient distinguisher that makes two distinct queries and outputs 1 if and only if the xor of the inputs is equal to the xor of the outputs. Clearly, if the values are the outputs of F_k, the attacker returns 1 with probability 1; if the values are taken from a truly random function R, then the attacker outputs 1 with probability equal to $2^{-\lambda}$. Then it follows:

$$\begin{aligned} &\mathbb{P}[A(y_1, y_2) : y_1 = F_k(x_1); y_2 = F_k(x_2), k \leftarrow_\$ \{0,1\}^\lambda] + \\ -\,&\mathbb{P}[A(y_1, y_2) : y_1 = R(x_1); y_2 = R(x_2), R \leftarrow_\$ \mathcal{R}(\lambda, \lambda + l)] \geq 1 - 2^{-\lambda} \end{aligned}$$

that is non negligible. □

[3] in the sense that computes $F_k(x) \forall k, x$.

(ii) $F_k(x) := F_x(k)$, where $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^{\lambda+l}$ is a PRF.
SOL This solution does not work. Assume the existence of a very bad key k_b that transforms F into a constant function (e.g. always 0): this subtle property does not alter the security of $F_k(\cdot)$ for a random key, since this bad key is only chosen with negligible probability. However, the proposed construction allows the attacker to fix the key (it is the input x passed by the attacker!), so in order to distinguish, he simply queries the challenger with input k_b: then it is easy to distinguish with non negligible probability, since the attacker returns 1 if and only if he obtains 0. In case of PRF, the attacker returns 1 with probability 1; otherwise it returns 1 with probability $2^{-\lambda}$. □

(iii) $F'_k(x) = F_k(k||0)||F_k(k||1)$, where $x \in \{0,1\}^{n-1}$.
SOL This solution works instead. Assume not: then there exists a PPT attacker $\mathcal{A}$ able to break $F'_k(\cdot)$. I show how to build on top of $\mathcal{A}$ an attacker $\mathcal{A}^*$ able to break the security of F. In particular $\mathcal{A}$ can distinguish $F'_k(\cdot)$ from a truly random function $R \leftarrow_\$ \mathcal{R}(n-1, n)$. Whenever $\mathcal{A}$ queries some input x, $\mathcal{A}^*$ forwards two queries $x_b := x||b$, for $b \in \{0,1\}$, after receiving the values y_1, y_2, sends back to $\mathcal{A}$ the value $y := y_1||y_2$. Finally, $\mathcal{A}^*$ returns the same bit b' of $\mathcal{A}$ and distinguishes as well as $\mathcal{A}$ does. Note that the number of queries to the challenger is the double of the queries needed by $\mathcal{A}$: however this is perfectly legitimate (it is still polynomial!). □

6 Secret-Key Encryption

(a) Prove that no secret-key encryption scheme $\Pi = (\texttt{Enc}, \texttt{Dec})$ can achieve chosen-plaintext attack security in the presence of a computationally unbounded adversary (which thus can make an exponential number of encryption queries before/after being given the challenge ciphertext).

SOL Without loss of generality we can say that any SKE scheme $\Pi = (\texttt{Enc}, \texttt{Dec})$, is such that:

$$c \leftarrow_\$ \texttt{Enc}(k, m)$$
$$m = \texttt{Dec}(k, c)$$

for $k \in \{0,1\}^\lambda$. Note that here I assume that the algorithm Enc uses the secret key k in addition to some source of random (e.g. nonces) in order to produce the ciphertext c: without loss of generality I assume that, once we fix the secret key and the message m, the maximum number of valid ciphertexts associated with m is some $p(\lambda) \in poly(\lambda)$ (clearly this quantity depends on the randomness injected by Enc); instead, as discussed in class, I assume that Dec is a deterministic algorithm.

I claim that the following unbounded attacker (see figure 4) $\mathcal{A}$ is able to distinguish with non negligible probability the CPA games[4] $G^{CPA}_{\Pi,\mathcal{A}}(\lambda, b), b \in \{0,1\}$: $\mathcal{A}$ sends two messages m^*_0 and m^*_1, and receives a ciphertext c^*_b (remind that the goal for the attacker is to guess b). Then it asks for exponentially many encryptions of m^*_0: if it gets back c^*_b, then returns $b' = 0$; otherwise, if at the end of all these queries, it never observes the desired ciphertext, returns $b' = 1$. Note that if the challenger selects $b = 1$, the attacker wins with probability 1 (because it will never[5] observe c^*_1 asking for the encryption of m_0); instead, if $b = 0$, it loses the game with negligible probability. Indeed, let assume that the number of queries is $Q = 2^{p(\lambda)}$: then, the probability that after Q runs, the challenger never encrypts the message in the same way[6] he did the first

[4] as defined in class
[5] Dec is deterministic: it uses the secret key and (part of) the ciphertext to extract the message.
[6] clearly the secret key k is fixed: what continuously changes is the random information used by Enc.

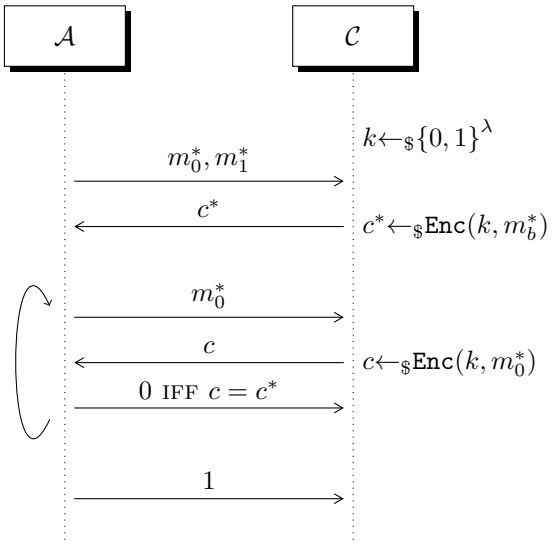

Figure 4: Simulation of $Game^{\mathtt{SKE\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda, b)$ for an unbounded $\mathcal{A}$ adversary that asks exponentially many times to encrypt the same message m_0^* trying to get a collision.

time is $\leq (1 - \frac{1}{2^{p(\lambda)}})^{2^{p(\lambda)}} \leq \frac{1}{e}$. But then:

$$\mathbb{P}[G^{CPA}_{\Pi,\mathcal{A}}(\lambda, 1) = 1] - \mathbb{P}[G^{CPA}_{\Pi,\mathcal{A}}(\lambda, 0) = 1] \geq 1 - \frac{1}{e}$$

i.e. non negligible. □

(b) Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$ be a family of pseudorandom permutations, and define a fixed-length encryption scheme $(\mathtt{Enc}, \mathtt{Dec})$ as follows: upon input message $m \in \{0,1\}^{n/2}$ and key $k \in \{0,1\}^\lambda$, algorithm $\mathtt{Enc}$ chooses a random string $r \leftarrow_{\$} \{0,1\}^{n/2}$ and computes $c := F_k(r||m)$. Show how to decrypt, and prove that this scheme is CPA-secure for messages of length $n/2$.

SOL

Since F is a PRP it is efficiently invertible; so in order to decrypt, we do: $F_k^{-1}(c) = r||m$ and then we only take the last $n/2$ bits.

Assume the scheme is not CPA-secure: then there exists an efficient attacker $\mathcal{A}$ able to break our scheme (win the CPA game with non negligible probability); but then we can break F too, i.e. we can build on top of it an attacker $\mathcal{A}^*$ able to distinguish with non negligible probability between a random distribution (figure 6) and the output of F_k (figure 5). Indeed, A^* whenever $\mathcal{A}$ asks for some message m to be encrypted, appends this to some random $r \in \{0,1\}^{n/2}$, and forwards this new message $m' := r||m$ to the challenger $\mathcal{C}$; finally, it forwards the ciphertext c, provided by $\mathcal{C}$ to the attacker. This is done for all the messages queried by $\mathcal{A}$ (this is allowed, of course, up to a polynomial number of queries since both $\mathcal{A}$ and $\mathcal{A}^*$ are PPT) both before and after the messages m_0^*, m_1^*: when these two messages are sent by $\mathcal{A}$, $\mathcal{A}^*$ selects at random a bit b, simulating the CPA game, modifies and forwards m_b^* in the very same way as before. When the attacker outputs a bit b', $\mathcal{A}^*$ forwards to the challenger 0 if and only if $b = b'$. Note that, if the challenger takes the value at random, we expect to win only in half of the cases; otherwise, by assumption, $\mathcal{A}^*$ returns 0 with a non negligible probability $\geq \frac{1}{2} + \frac{1}{p(\lambda)}$. This means that $\mathcal{A}^*$ can distinguish between the two scenarios (i.e. the two games) since:

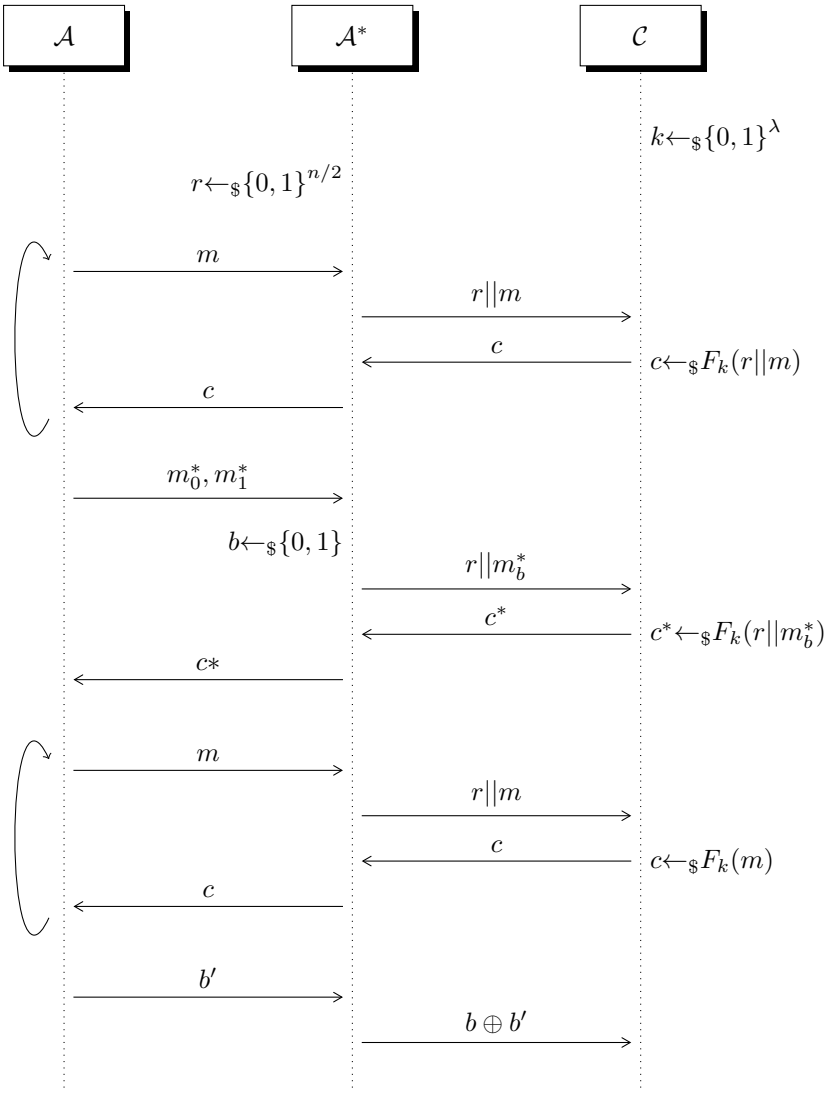

Figure 5: Simulation of $Game^{\mathtt{PRF}}_{\mathcal{A}^*}(\lambda)$. Base an attack to PRF F_k, assuming the existence of a PPT attacker $\mathcal{A}$, able to break the CPA-security of F with non negligible probability. $\mathcal{A}^*$ returns a bit b to the challenger $\mathcal{C}$, in order to distinguish.

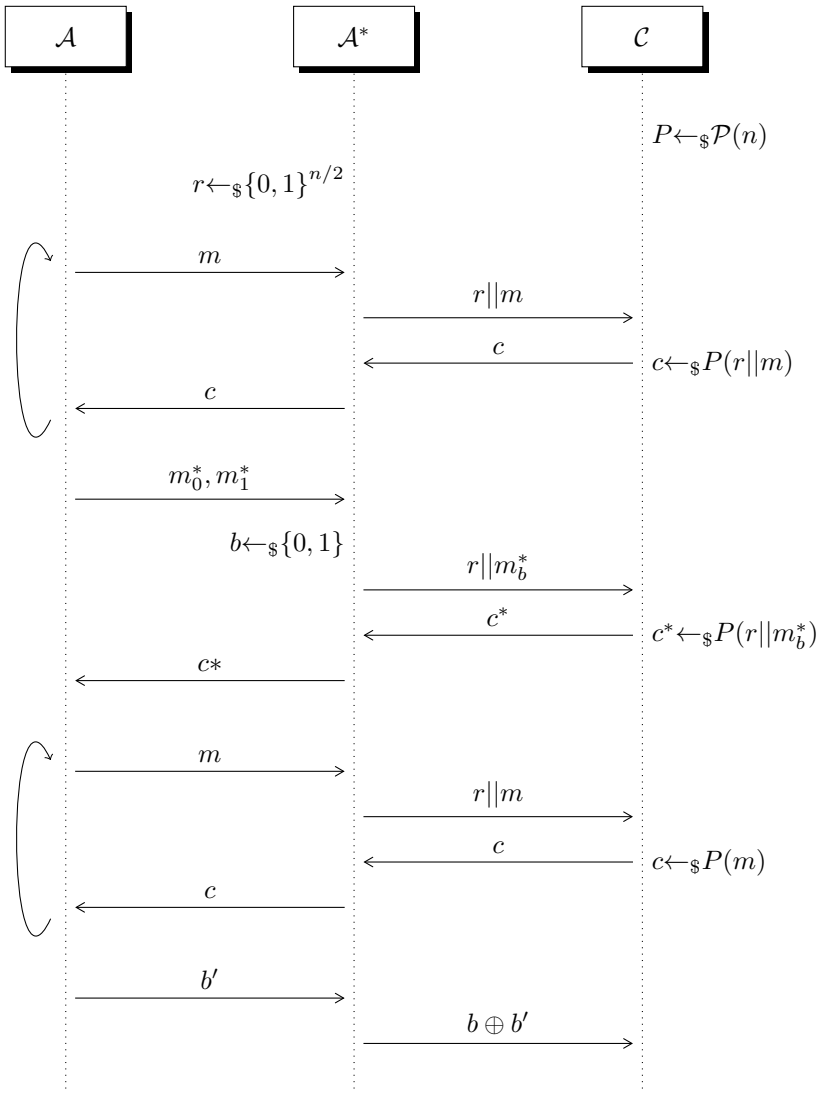

Figure 6: Simulation of $Game_{\mathcal{A}^*}^{\texttt{random}}(\lambda)$. Same approach of 5, but in this case the challenger uses random values from a randomly chosen permutation P.

$$\mathbb{P}[Game_{\mathcal{A}^*}^{\texttt{PRF}}(\lambda)=0]-\mathbb{P}[Game_{\mathcal{A}^*}^{\texttt{random}}(\lambda)=0]\geq\frac{1}{p(\lambda)}$$

that is non negligible. □

7 Message Authentication

(a) Assume UF-CMA MACs exist. Prove that there exists a MAC that is UF-CMA but is not strongly UF-CMA. (Recall that strong unforgeability allows the attacker to produce a forgery (m^*, τ^*) such that $(m^*, \tau^*) \neq (m, \tau)$ for all messages m queried.

SOL Under this assumption, we can always build M' such that $\texttt{Tag}'(m)$:=$(\texttt{Tag}(m)||b), b\leftarrow_{\$}\{0,1\}$ and $\texttt{Vrfy}'(m,\tau)$:=$\texttt{Vrfy}(m,\tau')$, where τ' is τ without the last bit: with this scheme, for each message m, we can produce $2k$ valid tags, for some $k \in \mathbb{N}\backslash\{0\}$

It easy to prove that M' is not strongly UF-CMA, because if we know a valid pair (m^*, τ_1), obtained querying the oracle, we can produce a new valid pair (m^*, τ_2), flipping the last bit of τ_1. But this scheme is still UF-CMA: assume not; then there exists some efficient attacker $\mathcal{A}$ able to produce, with non negligible probability, a valid pair (m, τ): this pair is still valid for M if we remove the last bit of τ. □

(b) Assume a generalization of MACs where a MAC Π consists of a pair of algorithms $(\texttt{Tag}, \texttt{Vrfy})$, such that Tag is as defined in class (except that it could be randomized), whereas Vrfy is a deterministic algorithm that takes as input a candidate pair (m, τ) and returns a decision bit $d \in \{0, 1\}$ (indicating whether τ is a valid tag of m). Consider a variant of the game defining UF-CMA security of a MAC $\Pi = (\texttt{Tag}, \texttt{Vrfy})$, with key space $K = \{0,1\}^\lambda$, where the adversary is additionally granted access to a verification oracle $\texttt{Vrfy}(k, \cdot, \cdot)$.

(i) Make the above definition precise, using the formalism we used in class. Call the new notion "unforgeability under chosen-message and verification attacks" (UF-CMVA).

SOL Let consider the following game $Game_{\Pi,\mathcal{A}}^{\texttt{UF-CMVA}}(\lambda)$:

i. the challenger $\mathcal{C}$ choses a key $k\leftarrow_{\$}\{0,1\}^\lambda$
ii. $\mathcal{A}$ can query (up to a polynomial number of times) $\mathcal{C}$ with some message m and receive back $\texttt{Tag}(k, m)$
iii. $\mathcal{A}$ can query (up to a polynomial number of times) $\mathcal{C}$ with some pair (m, τ) and receive back the output of $\texttt{Vrfy}(k, m, \tau)$
iv. $\mathcal{A}$ forges a pair (m^*, τ^*) such that $m^* \neq m, \forall m$ queried (m^* has to be "fresh")
v. the output of the game is $\texttt{Vrfy}(k, m^*, \tau^*)$ (i.e. 1 in case of win, 0 otherwise)

We say that a MAC Π is UF-CMVA-secure if $\forall$ PPT attackers $\mathcal{A}, \exists\epsilon(\lambda) \in negl(\lambda)$:

$$\mathbb{P}[Game_{\Pi,\mathcal{A}}^{\texttt{UF-CMVA}}(\lambda)=1]\leq\epsilon(\lambda)$$

□

(ii) Show that whenever a MAC has unique tags (i.e., for every key k there is only one valid tag τ for each message m) then UF-CMA implies UF-CMVA.

SOL First of all, note that the only difference between the two games (i.e. the two definitions of security UF-CMA and UF-CMVA) is that in $Game_{\Pi,\mathcal{A}}^{\texttt{UF-CMVA}}(\lambda)$ the attacker can verify the validity of pairs (m, τ) before submitting its proposal (m^*, τ^*). It is easy

to see that if for every key k, there is only one valid tag τ for each message m, the above situation can be simulated in a very simple way.

Whenever $\mathcal{A}$ asks for some pair (m, τ) to be verified, we can simply forward to the challenger m and receive back some τ_m: if $\tau_m = \tau$, then we know that $\texttt{Vrfy}(m, \tau) = \texttt{Vrfy}(m, \tau_m) = 1$; otherwise, we know[7] that the pair (m, τ) is not valid (i.e. $\texttt{Vrfy}(m, \tau) = 0$). Of course, this trick does not alter the number of queries (because $\mathcal{A}$ is a PPT making a polynomial number of queries) and does not modify the correctness. □

(iii) Show that if tags are not unique there exists a MAC that satisfies UF-CMA but not UF-CMVA.

SOL From previous exercise, we know that if tags are unique (once the key is fixed), then UF-CMA implies UF-CMVA. Instead, if this condition does not hold, we can build some UF-CMA scheme that is not UF-CMVA. Assume to have some UF-CMA-secure scheme $\Pi{:=}(\texttt{Tag}, \texttt{Vrfy})$, that produces tags of n bits for secret keys $\in \{0,1\}^\lambda$.

Let $\Pi'{:=}(\texttt{Tag}', \texttt{Vrfy}')$ such that $\texttt{Tag}'(k, m)$ computes $\texttt{Tag}(k, m)$ and appends some $x \in \{0,1\}^{\log \lambda}$ to it: this x is equal to $0^{\log \lambda}$, unless with some negligible probability, $2^{-\lambda}$, the algorithm $\texttt{Tag}'$ decides to chose as x the binary representation of an index i such that the i-th bit of the secret key $k_i = 1$[8]. The definition of algorithm $\texttt{Vrfy}'$ is straightforward, since it has to check the correctness of the first n bits of the tag, i.e. it uses $\texttt{Vrfy}$ on that part of the tag. Then it has also to check that the bit k_x is 1 (this is done if and only if $x \neq 0$).

Here I also provide the pseudocodes of such algorithms.

Pseudocode of $\texttt{Tag}'(k, m)$

Input: secret key k, message m
Output: a valid tag $\tau' \in \{0,1\}^{n+\log \lambda}$
$\tau \leftarrow_\$ Tag(k, m)$
luck $\leftarrow_\$ \{0,1\}^\lambda$
$x \leftarrow 0$
if $luck = 0^\lambda \wedge k \neq 0^\lambda$ **then**
  $x \leftarrow_\$ \{i : k_i = 1\}$
return $\tau || \langle x \rangle$[a]

[a] the binary ($\log \lambda$ bits) representation of x

Pseudocode of $\texttt{Vrfy}'(k, m, (\tau || x))$

Input: secret key k, message m, tag $(\tau||x), x \in \{0,1\}^{log\lambda}$
Output: a decision bit $d \in \{0,1\}$
$d \leftarrow \texttt{Vrfy}(k, m, \tau)$
$d' \leftarrow 1$
if $x \neq 0^{\log \lambda}$ **then**
  `/* the ∈ operator returns a bit (1 for true, 0 else) */`
  $d' = x \in \{\langle i \rangle : k_i = 1\}$
return $d \wedge d'$

It is clear that such a scheme cannot be UF-CMVA-secure, because an attacker $\mathcal{A}$, after receiving some valid pair $(m, (\tau||x))$ can start querying the challenger (up to $\lambda - 1$ times,

[7] we know it with probability 1: here it is crucial the assumption of unique tags, once the key is fixed.
[8] if such an index exists. The case for $k = 0^\lambda$ is also handled by the algorithm.

this is clearly polynomial) in order to leak bits of the secret keys: it tests all possible $x \in \{0,1\}^{\log \lambda}$, without varying m and τ; depending on the output of $\texttt{Vrfy}'$ on its inputs, it can understand whether a certain bit of k is 1 or 0. Once it knows $\lambda - 1$ bits of the secret key k, it is very simple to guess the last one (i.e. the first bit k_0): e.g. it can compute a tag (valid with probability 1/2) for a fresh message and ask the challenger to verify it; at this point it knows the full key, so it is really easy to always win the UF-CMVA game!

On the contrary, this scheme preserves the UF-CMA security property. Indeed, an attacker can only ask the challenger to tag messages, but has no control over the choice of the last $\log \lambda$ bits of the tag. Since the probability to **not** use $0^{\log \lambda}$ as "padding" is negligible ($2^{-\lambda}$ by definition of $\texttt{Enc}'$), we can assume that the attacker can observe one single[9] bit of the key only after exponentially (in λ) many queries. □

[9] the average number of bits of the secret key we expect to be leaked after 2^{λ} queries is 1: this is clearly not enough to retain a non negligible advantage, so the number of queries should be even higher to leak enough bits! But this is for sure out the possibilities of a PPT attacker.