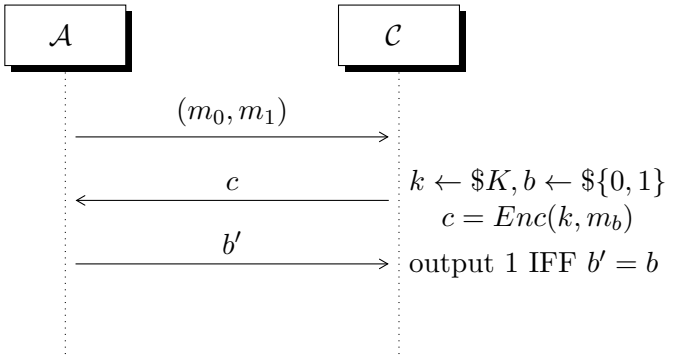
Cryptography Homework 1

(Solutions)

Giulio Ginesi - 1840066

Contents

1 Perfect Secrecy



From our hypothesis we have that (a): $Pr[Game^{eav}_{\pi,A} = 1] = \frac{1}{2}$ and we want to use the following notion of prefect secrecy (b): $Pr[Enc(K, m_0) = c] = Pr[Enc(K, m_1) = c]$.

We want to prove that $a \implies b$ and also $b \implies a$
For the first part $a \implies b$:

The notion (a) can be rewritten as

$$Pr[Game = 1|b = 0]Pr[b = 0] + Pr[Game = 1|b = 1]Pr[b = 1] =$$

Since b is chosen uniformly between 0 and 1 we have:

$$= \frac{1}{2}Pr[Game = 1|b = 0] + \frac{1}{2}Pr[Game = 1|b = 1] =$$

$$= \frac{1}{2}(Pr[Game = 1|b = 0] + Pr[Game = 1|b = 1])$$

(I give up)

2 Universal Hashing

2.1 point a

2.1.1 (i)

We can prove that $t - wise \implies (t-1) - wise$ just by writing down the notion of t-wise independence.

$$\sum_{i=0}^{t} Pr[h_s(x_i) = y_i] = \sum_{i=0}^{t-1} (Pr[h_s(x_i) = y_i]) + Pr[h_s(x_t) = y_t]$$

From the above equation we can see that the $t-1$-wise independence is contained in the t wise one.

2.1.2 (ii)

$$Pr[h_s(x_0) = y \wedge h_s(x_1) = y \wedge h_s(x_2) = y] = \frac{1}{|y^3|}$$

Now for 3 different x: $x_0 \neq x_1 \neq x_2$ the corresponding hash functions are as follows:

$$h_s(x_0) = S_0 + S_1 x_0 + S_2 x_0^2$$
$$h_s(x_1) = S_0 + S_1 x_1 + S_2 x_1^2$$
$$h_s(x_2) = S_0 + S_1 x_2 + S_2 x_2^2$$

$$Pr_{(S_0,S_1,S_2)\leftarrow Z_q^3}[h(x_0) = \varphi \wedge h(x_1) = \varphi\prime \wedge h(x_2) = \varphi\prime\prime] =$$

$$= Pr_{(S_0,S_1,S_2)\leftarrow Z_q^3}[S_0+S_1x_0+S_2x_0^2 = \varphi \wedge S_0+S_1x_1+S_2x_1^2 = \varphi\prime \wedge S_0+S_1x_2+S_2x_2^2 = \varphi\prime\prime] =$$

$$= Pr_{(S_0,S_1,S_2)\leftarrow Z_q^3}[\begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{pmatrix} \cdot \begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} \varphi \\ \varphi\prime \\ \varphi\prime\prime \end{pmatrix}] =$$

$$= Pr_{(S_0,S_1,S_2)\leftarrow Z_q^3}[\begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \varphi \\ \varphi\prime \\ \varphi\prime\prime \end{pmatrix}] =$$

$$= \frac{1}{|Z_q^3|} = \frac{1}{q^3}$$

2.2 part b

2.2.1 (i)

$l = 128$ what is the minimal min-entropy to achieve $\varepsilon = 2^{-80}$?

$$128 \leq k - (2log2^{80} - 2)$$

$$k \geq 286$$

Entropy loss: $\delta = 158$

2.2.2 (ii)

If $k = 238$ what is the maximum amount of uniform randomness with $\varepsilon = 2^{-80}$?

$$l \leq 238 - (2log2^{80} - 2)$$

$$l \leq 80$$

Making computational assumptions explain how to obtain $l = 320$
Let's assume a large enough $\varepsilon = 2^{-128}$ then our k will be:

$$320 \leq k - (2log2^{128} - 2)$$

$$k \geq 574$$

How can we obtain $l = 320$ using computational assumptions?

$$320 \leq k - (2log(\frac{1}{\varepsilon}) - 2)$$

I assume a large enough $\varepsilon = 2^{-128}$

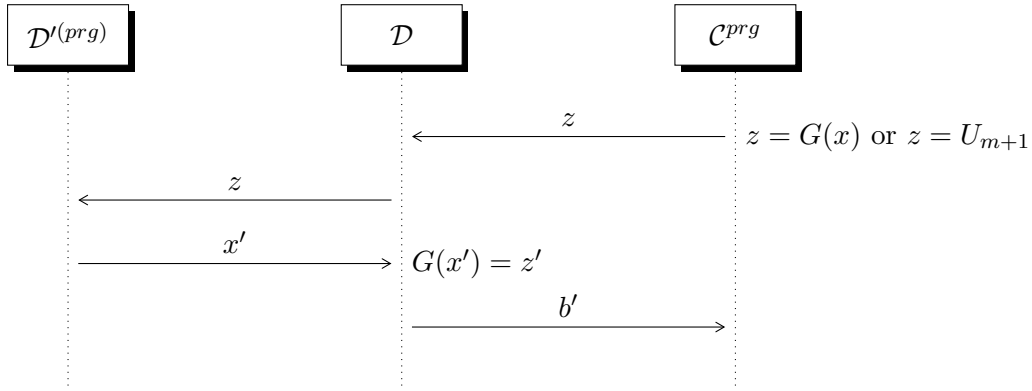
$$320 \leq k - (2 * 128 - 2)$$

$$320 \leq k - 254$$

$$k \geq 574$$

3 One-Way Functions

3.1 part a



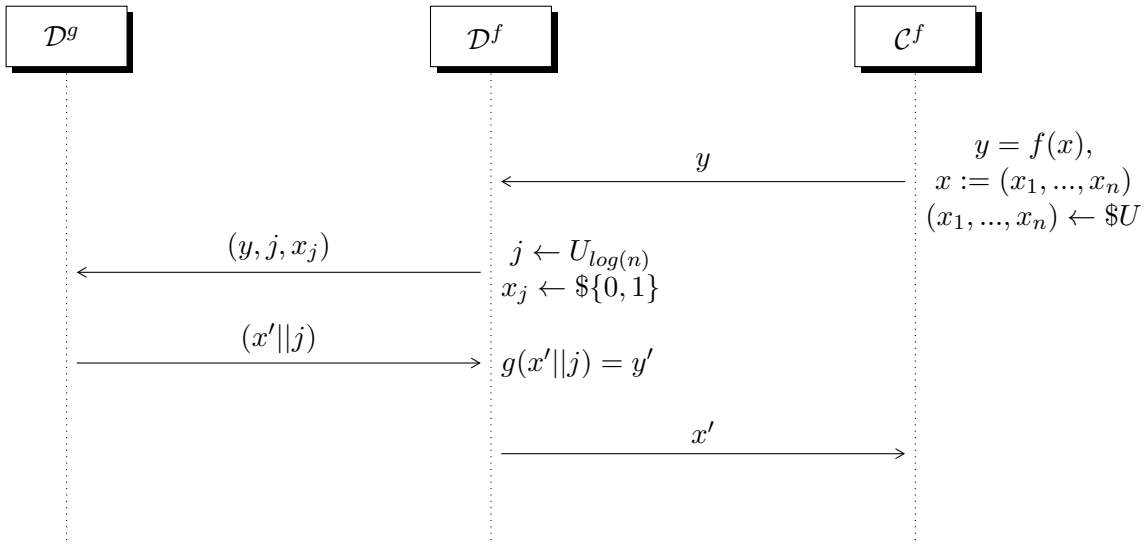
Suppose D' is able to invert G with non negligible probability, then D can distinguish in poly time if z comes from G or U_{n+1}. Upon receiving x' D can compute $G(x') = z'$. Now if $z' = z$, D will decide $b' = 1$ otherwise $b' = 0$.

There is one case in which D makes the wrong guess. Suppose that C chooses z from uniform and this z is exactly the output of $G(.)$ with a particular x' which is exactly the value given by D' when it receives $z \leftarrow \$U_{m+1}$. This is the probability of choosing a "colliding z" times the probability of D' giving as x' exactly the right value for which $G(x') = z$ but $z \leftarrow \$U_{m+1}$. This happens with $Pr = \frac{1}{|z|}\frac{1}{|x'|} = \frac{1}{2^{m+1}}\frac{1}{2^m} = \frac{1}{2^{2m+1}}$

3.2 part b

3.2.1 (i)

Idea: "Breaking g implies breaking f"



By contradiction assume that $\exists$ a distinguisher D^g that is able to invert g with non negligible probability. We want to use D^g to break C^f, since x_j is a bit and is chosen at random by D^f it will have $Pr = \frac{1}{2}$.

The probability that D^f is able to recover a valid x' from D^g is the probability that D^f chooses the right x_j (event C) and the probability that D^g returns a couple $(x'||j)$ such that $g(x'||j) = (y, j, x_j)$ (event D). Then

$$Pr[C \wedge D] = \underbrace{Pr[C]}_{\frac{1}{2}} \overbrace{Pr[D|C]}^{\geq negl(\lambda)} \geq negl(\lambda)$$

3.2.2 (ii)

Fix i, we construct an A_i s.t. $A_i(f(x\prime))$ outputs $x\prime_i$ with $Pr = \frac{1}{2} + \frac{1}{2n}$.
Now we can distinguish 2 cases:

1. For A_i where $i \in \{n+1, n+log(n)\}$, since A_i can see $(f(x), j, x_j)$ he can see also j and since i is in that interval. A has directly the correct value to output.

2. For A_i where $i \in \{1, n\}$, he can take x_j and hopes that $j = i$.

If we are in the first case $Pr[WIN] = 1$.
Otherwise:

$$Pr[WIN] = Pr[j = i] + Pr[j \neq i] =$$

$$= \frac{1}{n}1 + \underbrace{(1 - \frac{1}{n})}_{Pr[j^{th}\text{index} \neq i]} \overbrace{\frac{1}{2}}^{Pr[j^{th}\text{bit} = i^{th}\text{bit}]} =$$

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

4 Pseudorandom Generators

4.1 point a

The two functions G_1 and G_2 have two different distributions over $\{0,1\}^{\lambda+l}$, since one of the two is indistinguishable from uniform distribution while the other one is not.

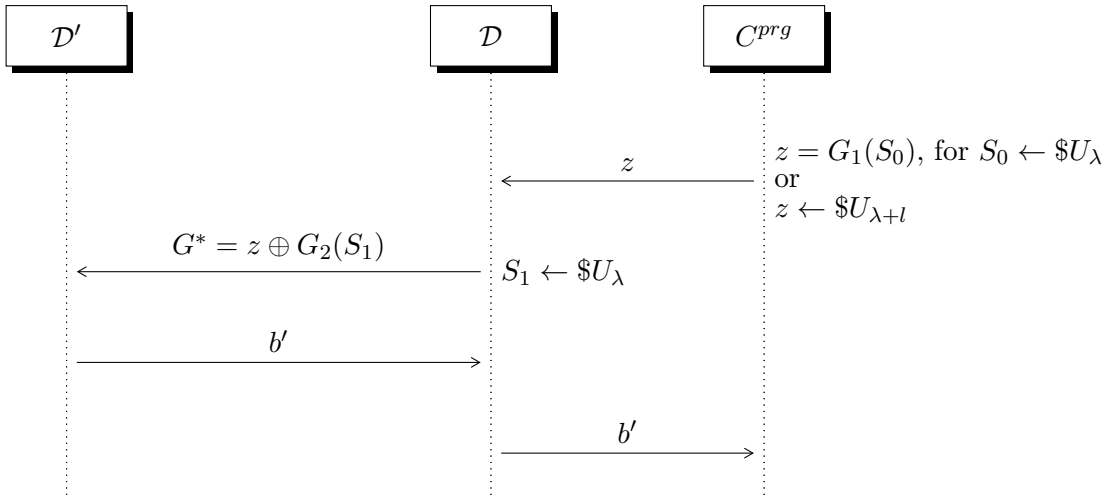
Intuitively we can state that the XOR operation preserves the distribution, and therefore the randomness (as we have done when demonstrating OTP). For this reason $G_1(.) \oplus G_2(.)$ has uniform distribution if one of the two is uniformly distributed.

For our problem let's create our new function G^* as:

$$G^*(S_1, S_2) = G_1(S_1) \oplus G_2(S_2)$$

where the seeds (S_1, S_2) are sampled at random over λ and $S_1 \neq S_2$.

Let's fix G_1 as our PRG without loss of generality, suppose we can build a PPT distinguisher which is able to tell apart G_1 and $U_{\lambda+l}$:



Now to distinguish between G^* and $U_{\lambda+l}$ means to distinguish between G_1 and $U_{\lambda+l}$ (for the xor property discussed before). Hence this is a contradiction since G_1 is a PRG by definition.
The exact same proof can be repeated if G_2 is a PRG.

4.2 point b

We would like to optimize the seed length (i.e. use the same seed for both functions). On this point I have two considerations:

1. The same seed can't be used since, looking at the above game, it would mean that D must have access to the seed S used by C. This is a big problem because with access to this information D can compute both $G_1(S)$ and $G_2(S)$ (the PRGs are public) and therefore will win the Game with probability=1 always.

2. There exists an example using a specific construction of $G_1 = G_2 \oplus 1^\lambda$ where G_1 is a PRG and therefore also G_2 is a PRG. Hence $G^*(S) = G_2(S) \oplus 1 \oplus G_2(S) = 1$ which is not a PRG.

Intuition: fixed G_1 as the PRG: $G^* = G_1(S_0) \oplus G_2(S_0) \oplus G_2(S_1)$ this a PRG even when using always the same seed ($G^* = G_1(S)$) however the problem discussed in 1) remains.

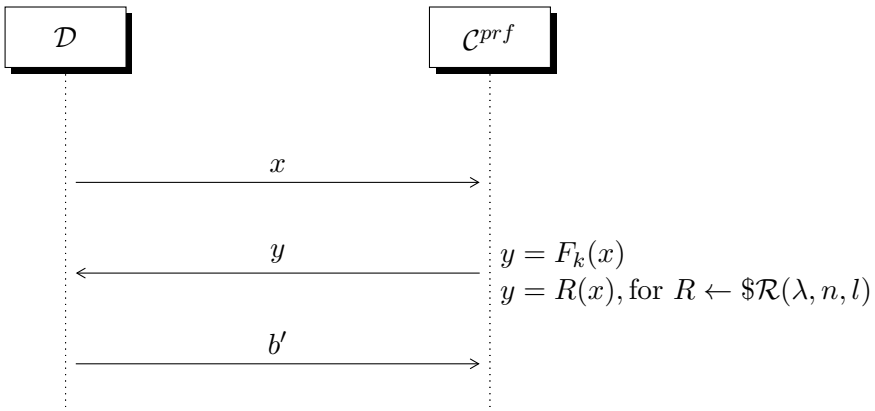
5 Pseudorandom Functions

5.1 point a

Consider an exponential time/unbounded distinguisher D that,

$$\forall \mathcal{F} = \{F_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{l(\lambda)}\}_{k \in \{0,1\}^\lambda}$$

plays the PRF game with this family of functions :



When $\mathcal{D}$ receives y, since he is unbounded, he can compute all the possible $F_k(x), \forall k \in \{0,1\}^\lambda$, in this way he can efficiently win the *Game* for each family of PRF functions.

Note 1: there could be sort of "collisions" when $\mathcal{C}$ chooses $y = R(x)$ but this can also be the result of $F_{k'}(x)$ but not of $F_k(x)$(where k is the key used in the PRF experiment), for a certain $k' \in \{0,1\}^\lambda$ and $k' \neq k$.
Then, $\mathcal{C}$ sends y to $\mathcal{D}$, which gives the wrong answer thinking that $\mathcal{C}$ is using $F_{k'}(x) = y$ while instead it is actually using R.
Thus, after sending the related b' to $\mathcal{C}$, $\mathcal{D}$ loses.

There are, in the worst case, $2^\lambda - 1$ possible common values among $F_{(.)}(x)$ and $R(x)$, the number of "hidden" keys minus the "legal" key chosen for the PRF experiment. So, the probability that $\mathcal{C}$, using a random function, extracts a truly random number and not an $y = F_{k'}(x)$ for some k' "hidden" key is $\frac{1}{2^l - (2^\lambda - 1)}$.

Note 2: It can happen that $\mathcal{C}$ chooses an $y = R(x)$ equal to $F_k(x)$. Like before, in this case $\mathcal{D}$ thinks he is playing with a PRF, while he is actually

playing with a truly random function, so $\mathcal{D}$ loses.

This is the probability of picking an input x over $|\mathcal{M}|$ possible inputs,times the probability of picking a Random function $R(.)$ which contain all the possible random functions.
Since the random functions with a table containing a collision with F_k over input x are $2^{l2^{n-1}}$, the probability of picking exactly x and one of these random functions containing a collision over x is

$$\frac{1}{2^n} * \underbrace{\frac{2^{l2^{n-1}}}{2^{l2^n}}}_{\frac{1}{2}}$$

which is negligible.

5.2 point b

5.2.1 i

Suppose that the proposed $F_k(.)$ is a PRF. This means that $F_k(x)$ for any x is indistinguishable from $R(x)$, for $R \leftarrow \$\mathcal{R}(\lambda, n, l)$.
Now, given a couple (x, x') such that $x \neq x'$, it should be *difficult* to guess $z = R(x) \oplus R(x')$, and *difficult* means that, since there exists only one z which is the output of the XOR operation, it's possible to pick the correct one with probability $\frac{1}{2^l}$.
Otherwise, for a given random couple (x, x'), our function produces

$$\begin{gathered} F_k(x) = x \oplus G'(k) \wedge F_k(x') = x' \oplus G'(k) \Rightarrow \\ \Rightarrow F_x(x) \oplus F_k(x') = \\ x \oplus G'(k) \oplus x' \oplus G'(k) = \\ = x \oplus x' \end{gathered}$$

and it is possible to correctly guess this result with probability 1. So the proposed $F_k(.)$ is not totally indistinguishable from a truly random function.

5.2.2 ii

To prove that $F_k(x) = F_x(k)$ given is PRF, we should show that this property holds for every possible PRF F.
Now consider a function F' with the same definition of F, but when F' receives $k = t$ for a fixed $t \leftarrow \$\{0,1\}^\lambda$ it always outputs 1^l, no matter of what

is given as second argument.

Since F' behaves like F but for inputs t, it is indistinguishable from R until one yields one of these $t-$strings. Since among all the strings of $\lambda + \lambda$ bits there are λ strings in which the first part is exactly t,

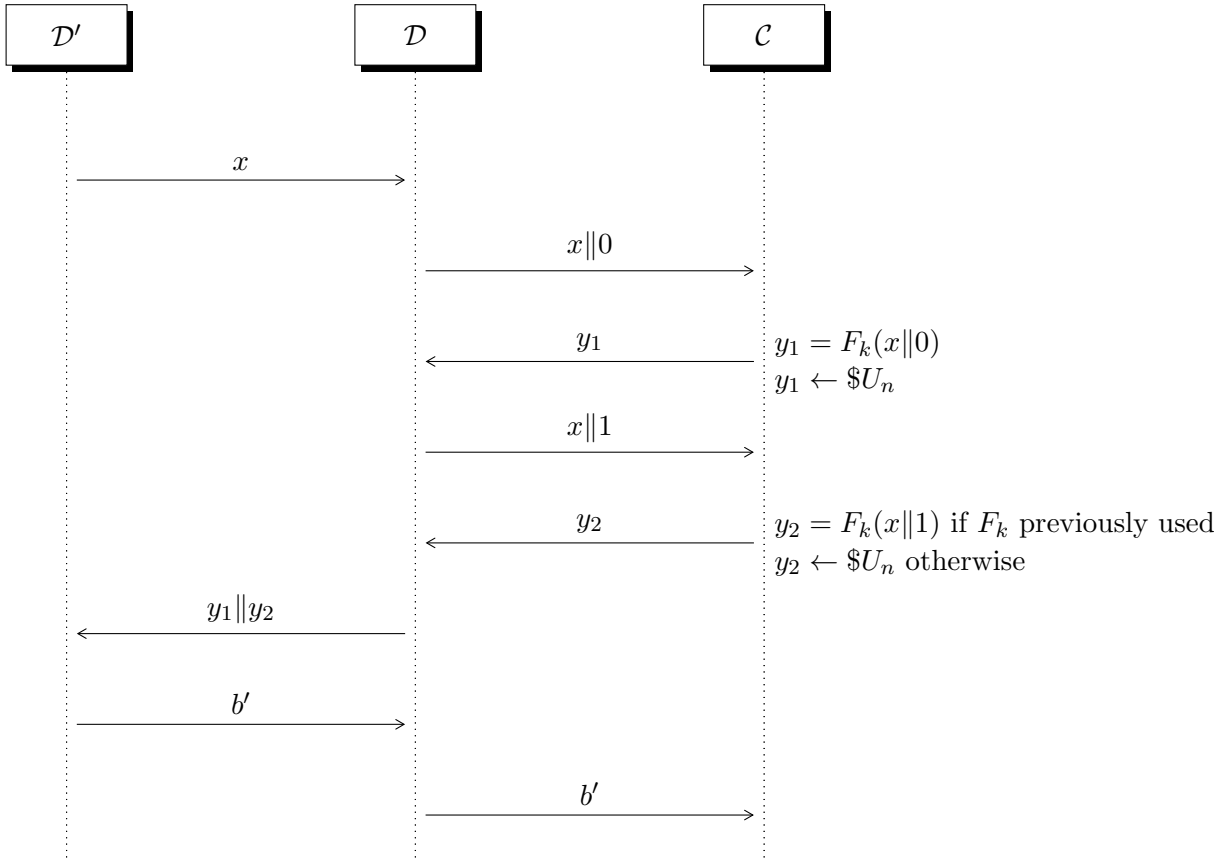
$$\mathcal{P}[\text{ Pick at random a } \lambda + \lambda \text{ string "starting" with } t] = \frac{1}{2^{\lambda}}$$

so F' is distinguishable from R for the same factor as F and for a negligible factor related to picking the "crashing strings"(defined before) at random. So we can say that F' is a PRF.

But now if an attacker sends t to the challenger who uses $F_k(x) = F_x(k)$, the attacker can guess with probability 1 if he is playing in an experiment involving a random function or not, since if the received message from the challenger is 1^l the function used is F' (less than a negligible factor, which is the probability that the random function picks exactly 1^l for the given input).

5.2.3 iii

Let's suppose that , given the PRF $F : \{0,1\}^n \to \{0,1\}^n$, there exists a distinguisher D' which can distinguish $F' : \{0,1\}^{n-1} \to \{0,1\}^{2n}$ from a truly random function $R : \{0,1\}^{n-1} \to \{0,1\}^{2n}$.
But if this is possible, then we have:

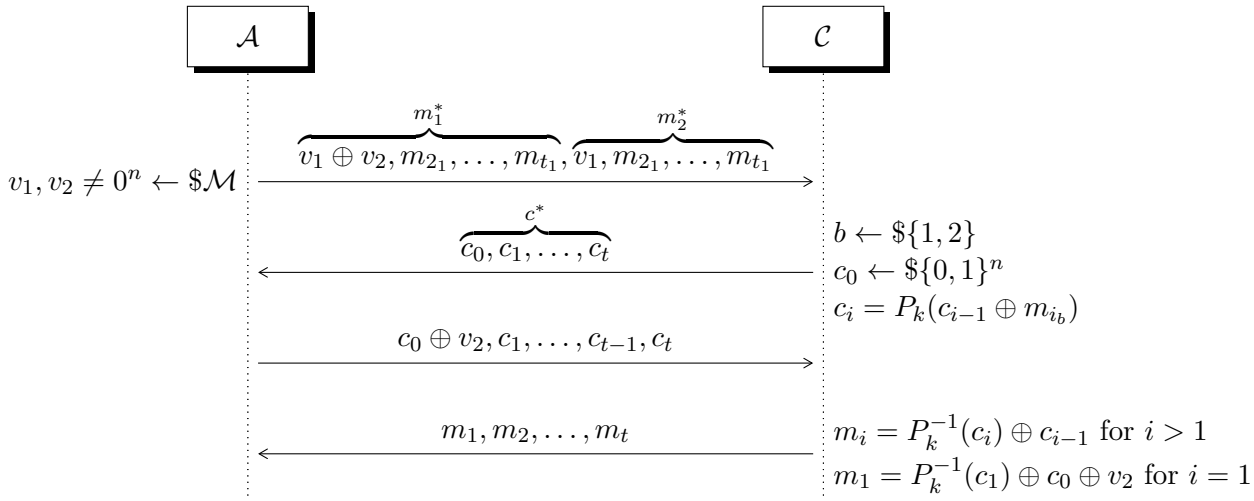


so we could break the PRF function F_k, which is impossible by definition.

6 Secret Key Encryption

6.1 point a

Consider the following counterexample:



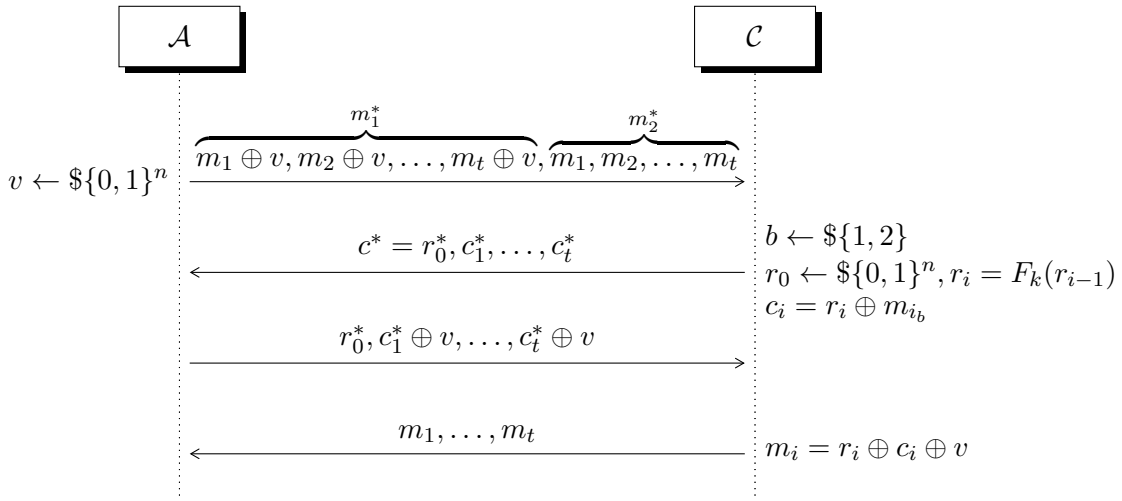
But $P_k^{-1}(c_1) = c_0 \oplus m_{1_b}$. So:

- if $b = 1$, the received m_1 contains $c_0 \oplus v_1 \oplus v_2 \oplus c_0 \oplus v_2 = v_1$
- if $b = 2$ the received m_1 contains $c_0 \oplus v_1 \oplus c_0 \oplus v_2 = v_1 \oplus v_2$

So looking in the first block of the last query we can understand which block was chosen as first, and we can deduce which message was encrypted since m_1^* and m_2^* share all the blocks except the first.

6.2 point b

Consider the following counterexample:



Now there are 2 possibilities:

- if $b = 1$, $m_i = r_i \oplus \overbrace{r_i \oplus m_i \oplus v}^{c_i} \oplus v = m_i$
- if $b = 2$, $m_i = r_i \oplus \overbrace{r_i \oplus m_i}^{c_i} \oplus v = m_i \oplus v$

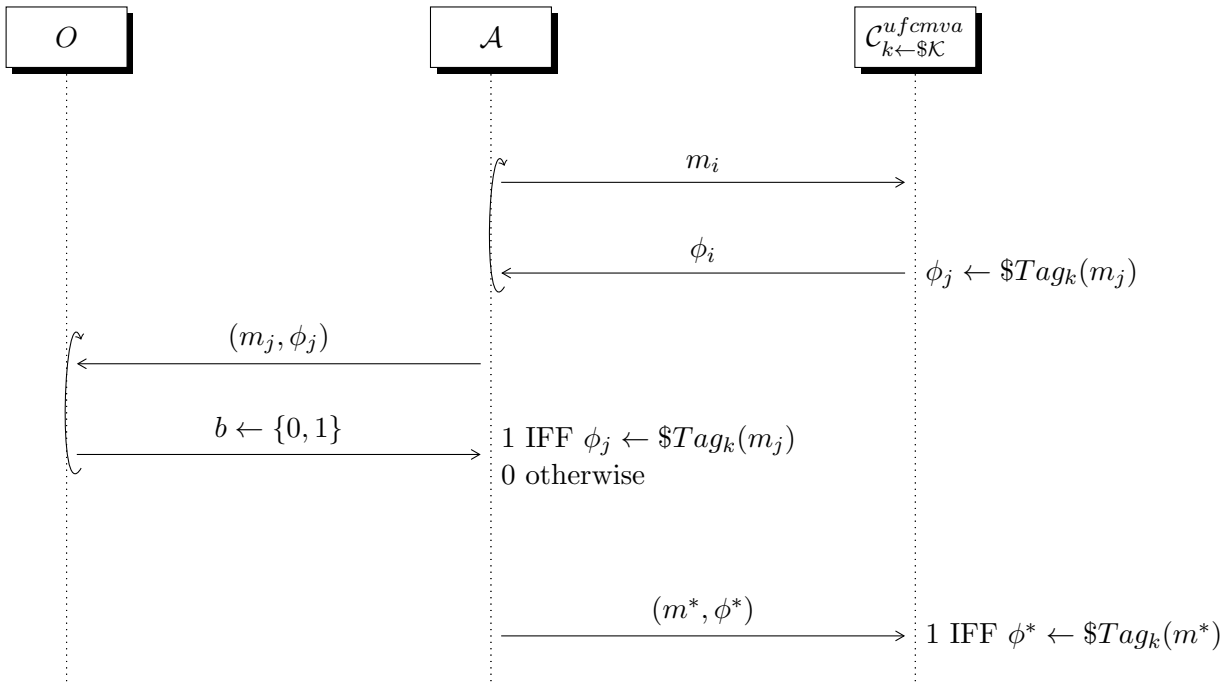
and this allows the Attacker to understand which message was chosen for encryption.

7 Message Authentication

7.1 point a

7.1.1 i

It's possible to define a **UFCMVA** property for a MAC scheme $\Pi = (Tag, Vrfy)$ as the following one:



So, if a MAC scheme is UFCMVA, the following formula must be valid for an attacker:

$$\mathcal{P}[Game^{ufcmva} = 1] \leq negl(\lambda)$$

There are few important considerations to make:

- $Tag_k(.)$ is supposed to be a randomized function, so it returns not a single value, but a set of possible values
- requests to oracle and to challenger from $\mathcal{A}$ can happen in any possible order (not necessarily as shown in the image)
- the last message is the challenge couple (m^*, ϕ^*), which wins only if the re-computed tag is exactly the same as ϕ^*

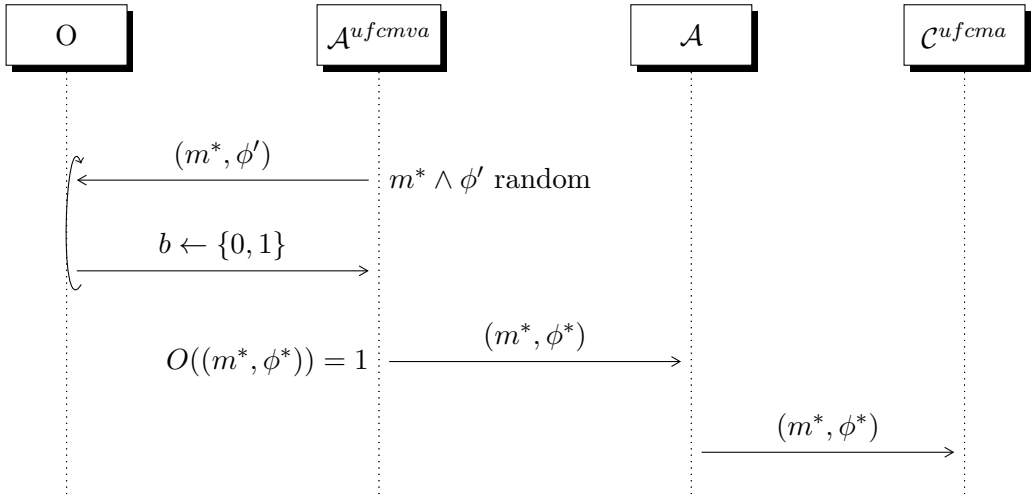
- the oracle tells $b = 1$ if ϕ_j is in the set of possible values computed by $Tag_k(m_j)$
- since C will have his own verification oracle, identical to O the couple (m^*, φ^*) must be fresh for C. Meaning that φ^* should not be an output of C for a non fresh m^*.

7.1.2 ii

To show that:

$$Tag \text{ has unique MACs} \wedge UFCMA \Rightarrow UFCMVA$$

let's prove it by reduction, supposing that an $\mathcal{A}^{ufcmva}$ adversary who can break $Game^{ufcmva}$ exists:



and then $\mathcal{A}$ wins, but this is a contradiction.

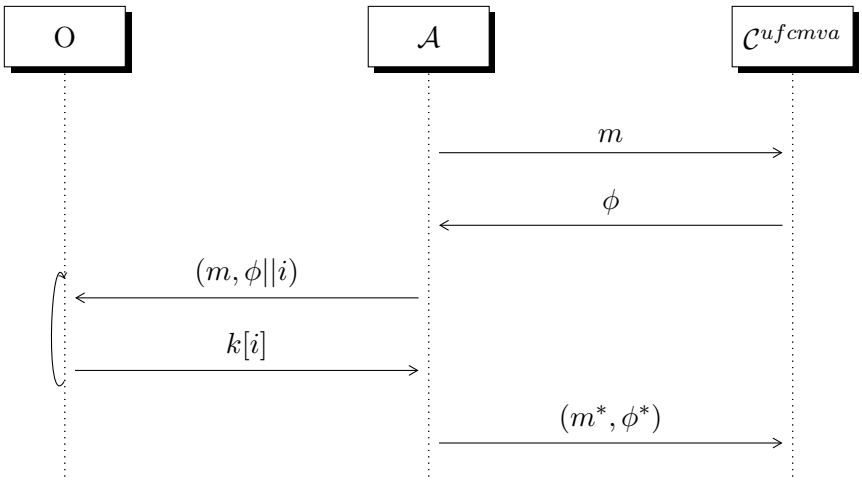
Note: the last message $\mathcal{A}$ sends to $\mathcal{C}^{ufcma}$ should be fresh (never asked to $\mathcal{C}^{ufcma}$), because of the standard definition of $Game^{ufcma}$. The only way that A can win, since m^* must be fresh, is just if A^{ufcmva} makes queries to the O fixing m^* and forging multiple Tags until he finds the right ϕ^* for which $b = 1$.

7.1.3 iii

Given $\Pi = (Tag, Vrfy)$, let's construct our $\Pi' = (Tag', Vrfy')$ as:

- $Tag'(x) = Tag(x) \| 0$
- $Vrfy'(k, m, \phi')$ executes:
  1. Split ϕ' into 2 parts: $\phi \| i$.
  2. Use the first part as input for $Vrfy(k, m, \phi)$ and store the output into d.
  3. Now if d=1 check if $i \geq 1$ and if so output $k[i]$, the i-th bit of the key; otherwise, simply return d.

Since $Tag'(.)$ does not change the original $Tag(.)$ function, this is till ufcma secure.
But now, for the following counterexample we can show that Π' is not ufcmva secure.

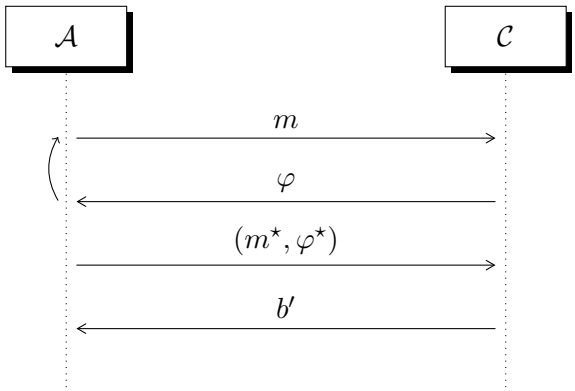


Given the $Vrfy'$ algorithm, now $\mathcal{A}$ obtains a valid tag ϕ for m and makes $|k|$ (length of the key k) queries to the oracle, to leak at every query a specific bit of the key.
In particular, every new query to the oracle contains an incremental value of i until $|k|$, so the key is completely discovered after $|k|$ queries.

In the end, $\mathcal{A}$ simply chooses a fresh random message m^*, uses the key for computing $Tag_k(m^*) = \phi^*$ and sends this couple to the challenger, being sure of winning.

7.2 b

Consider the following game form a generic CBC-MAC:



7.2.1 (i)

CBC-MAC for VLM is not secure. Consider the following construction:

Obtain a message $m_0 \leftarrow \$\{0,1\}^n$ of 1 block and his Tag $\varphi_0 = F_k(m_0)$ I can construct a message $m^\star = (m_0, m_0 \oplus \phi_0)$ and a valid Tag $\varphi^\star = \varphi_0$. In this way when the challenger will verify the message he will compute $\varphi^{\star\prime} = F_k(m_0 \oplus \varphi_0 \oplus \varphi_0) = F_k(m_0) = \varphi_0$. And then clearly $\varphi^\star = \varphi^{\star\prime}$

7.2.2 (ii)

CBC-MAC using randomness as initialization vector. This isn't secure, consider the following construction:

Obtain 2 messages m_0, m_1 with $m_0 \neq m_1$ and their tags $\varphi_0 = (r_0, F_k(m_0 \oplus r_0)), \varphi_1 = (r_1, F_k(m_1 \oplus r_1))$. I can now forge a new message $m^\star = (m_0 \oplus m_1)$ and a valid Tag $\varphi^\star = (r_1 \oplus m_0, \varphi_1)$. When the challenger will verify $\varphi^\star$ he will compute $\varphi^{\star\prime} = F_k(m_1 \oplus m_0 \oplus r_1 \oplus m_0) = F_k(m_1 \oplus r_1) = \varphi_1$ therefore $\varphi^{\star\prime} = \varphi^\star$.

7.2.3 (iii)

CBC-MAC that outputs the tag for each block $\varphi = \varphi_1, \varphi_2, ..., \varphi_t$ is not secure. Consider the following construction:

Obtain 2 messages $m_0 = (m_{0,1}, m_{0,2}), m_1 = (m_{1,1}, m_{1,2})$, of at least 2 blocks, and their tags $\varphi_0 = (\varphi_{0,1}, \varphi_{0,2}), \varphi_1 = (\varphi_{1,1}, \varphi_{1,2})$. Now i can construct $m^\star = (m_{1,1}, m_{2,1} \oplus \varphi_{1,1} \oplus \varphi_{2,1})$ and a valid Tag $\varphi^\star = (\varphi_{1,1}, \varphi_{2,2})$.

When the challenger will verify $\varphi^\star$ he will compute $\varphi^{\star\prime} = F_k(m_{1,1}), F_k(m_{2,1} \oplus \varphi_{1,1} \oplus \varphi_{2,1} \oplus \varphi_{1,1}) = \varphi_{2,2}$ therefore $\varphi^\star = \varphi^{\star\prime}$.